Technical Report

DomainTools

# Speeding Up DNSDB Queries via Parallelization

Version 1.1

**Joe St Sauver, Ph.D.**
Distinguished Scientist

# Contents

# Executive Summary

DomainTools Farsight DNSDB Passive DNS API allows subscribers to run up to ten concurrent query streams. Many DNSDB users, however, simply make serial queries over a single connection, resulting in unnecessary delays and lower-than-possible throughput. This report takes an iterative approach to showing how DNSDB subscribers can easily use parallel query streams to make their query workload run faster, covering both `dnsdbq` and DNSDB API (accessed via Python3).

This report takes an incremental approach to showing users how to parallelize their queries, starting simply and then exploring more complex approaches. We begin with sequential ("serial") approaches, and then make small modifications to "go parallel." We evaluate three different models for distributing domains for Python3 multiprocessing `(pool.imap_unordered, pool.map, and pool.imap)`. **All three parallel multiprocessing paradigms worked well (as did threaded and asyncio-based approaches), and all delivered a significant speedup.**

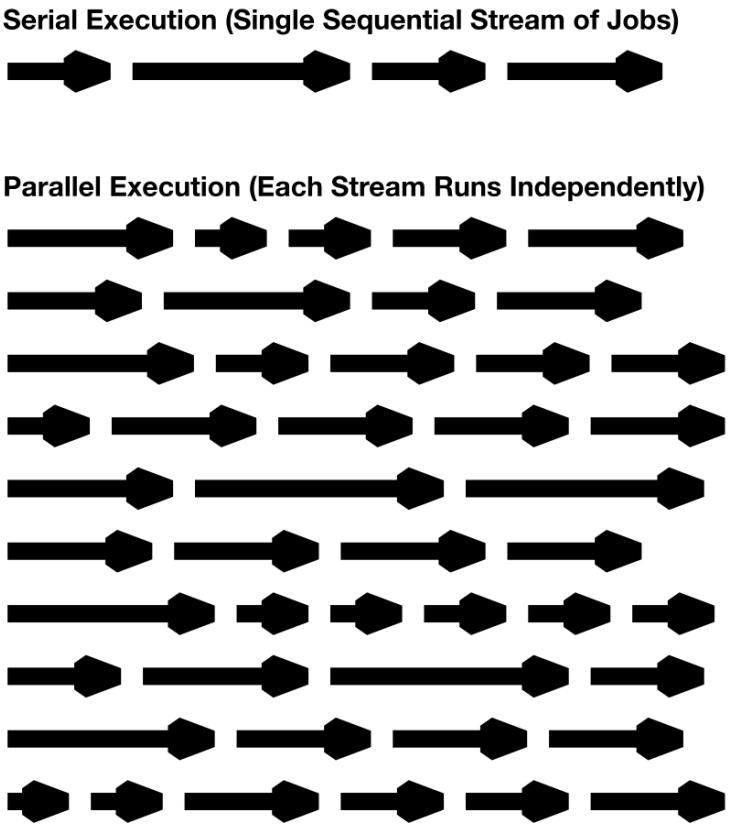
While doing our testing, we uncovered a number of other surprising phenomena we hadn't expected, including:

- DNSDB API appears to have a propensity to cache some results server-side; this appears as a small-but-noticeable increase in the number of returned results when the same queries are repeatedly re-run.
- The Python3 requests library supports a "session pool" that allows queries to the same server to be reused by subsequent queries. This can significantly accelerate https session traffic. Unfortunately, the requests sessions feature may not be thread safe, and hence was not used for our testing.
- While the MacBook Pro has a GPU capable of delivering over 2.5TFLOP, and an NPU capable of delivering 15.8 TFLOPS, those specialized processors may not be readily available for tasks such as our test workload.

# Introduction

The DomainTools Farsight DNSDB Passive DNS API (Farsight Security is now part of DomainTools) allows subscribers to run up to ten concurrent ("parallel") query streams. Many DNSDB users, however, just make sequential queries over a single connection, resulting in unnecessary delays and lower-than-possible throughput. This report takes an iterative approach to explaining how DNSDB subscribers can use parallel query streams to make their query workload run faster, covering both `dnsdbq` and using DNSDB API via Python3.

The easiest way to explain the difference between serial and parallel execution paradigms may be with an illustration:




Serial execution is what you get by default when running DNSDB passive DNS queries with:

- `dnsdbq`，the company's command line DNSDB client, at least as normally used
- DNSDB Scout，the company's web interface to DNSDB, or

- most 3rd-party DNSDB integrations.

When doing serial queries, a DNSDB query gets made, results get found and returned, and after that query's done, another query can then run. This is very straightforward BUT *NOT* particularly efficient.

Parallel execution runs multiple streams of jobs concurrently – in DNSDB's case, up to ten simultaneous streams of queries for each subscriber. Running multiple jobs in parallel potentially delivers dramatically improved throughput: you'll get more work completed in less time.

In fact, in this case, we have what's referred to as an "embarrassingly parallel" problem that we can easily "divide up" and run chunk-by-chunk with no interaction between the chunks – perfect for parallel execution paradigms.

# Organization of The Remainder of the Document

Since this is a relatively long document, let's now quickly describe how the rest of this document is organized.

**In Part I,** we begin by looking at the company's **command line client, `dnsdbq`**, which supports both interactive and "batch mode" queries. For `dnsdbq`, we'll make:

- A single `dnsdbq` query (just to show how `dnsdbq` routinely gets used interactively by customers)
- Then we'll show you how to use `dnsdbq -f` to make **sequential queries** in dnsdbq **batch mode**
- And finally, we'll use `dnsdbq -fm` to make **parallel queries** in dnsdbq **batch mode**. Parallel batch mode queries represent a really FAST option for the *non-programmer* who may nonetheless want to make his or her queries run as quickly as possible.

**In Part II,** we'll access the **DNSDB API from Python3, serially.** We'll start simply and then incrementally increase the complexity level:

- *Single Minimal Query:* We'll begin by making a single query with DNSDB API from Python3, just to get a sense of what's involved in making ANY sort of DNSDB API query from Python3.
- *Serial Python3 Code:* We'll then create sample code that makes a series of DNSDB API queries one after another from Python3 as a throughput baseline. This will also let us introduce some Python3 utility functions we'll need for both this baseline program and subsequent parallel iterations.

**In Part III,** we'll access **DNSDB API from Python3 in parallel,** trying several approaches:

- *Parallel Multiprocessing:* We begin our review of parallel approaches with Python3 multiprocessing. Believe it or not, taking advantage of parallel multiprocessing requires only minor changes to our sequential code, and parallel multiprocessing delivers a huge improvement in throughput. We'll evaluate three different models for distributing domains for parallel processing: `pool.imap_unordered, pool.map, and pool.imap.` All work well.
- *Parallel Threading:* Because of the Python3 Global Interpreter Lock ("GIL"), Python3 threads run on a single CPU. In this case, however, because of the nature of our workload, we find Python3 threading runs roughly as fast as Python3 multiprocessing.
- *Parallel Asyncio:* Next, we'll show you how we can make parallel DNSDB API queries using asyncio in Python3. This can be as fast as Python3 multiprocessing and threading code, and is a more flexible alternative that some may prefer, although the overall complexity of asyncio code (and the associated opportunities for mis-steps) tend to be somewhat greater.

- *Parallel Cythonized Asyncio:* Then we'll try Cython to convert the interpreted Parallel Asyncio code into a compiled executable. For some compute-intensive workloads, running relatively slow interpreted (rather than compiled) code can indeed be a source of overall poor code slowness. We'll empirically test the "compiled is faster than interpreted" notion for our workload by compiling and benchmarking one of our parallel codes with Cython.
- *Growth in Result Count?* We'll wrap this section up by doing 200 runs of our parallel asyncio code to see if the growth in result count we observed incidentally appears to ever "plateau."

**In Part IV**, we'll consider some often-promising **alternatives that actually turn out to *not* be particularly helpful when it comes to improving DNSDB API throughput**:

- Since we're downloading a significant number of results, a natural thought is, "Perhaps we could **enable compression** to reduce the size of our downloads?" Unfortunately, the DNSDB API server does not currently support compression.
- The Macbook Pro has both **four "Performance" and four "Efficiency" CPU cores.** Would forcing our work onto the faster cores help speed things up? The Macbook Pro lso has **8 Graphic Processing Unit ("GPU") cores as well as a 16 core "neural engine"** which would normally be a target for parallelization. Sadly our work is not the math-heavy workload that's well suited to those processors.

We conclude in **Part V** with a summary of what we've learned. Our Python3 code, our R code, and the list of dot edu domains we tested are all included in appendices.

# PART I: dnsdbq

(https://github.com/dnsdb/dnsdbq)

> ***"It is in starting with the first step that other steps become clearer."***
>
> ***Israelmore Ayivor, Leaders' Frontpage***

## (1) Using dnsdbq Interactively

After installing `dnsdbq` from `https://github.com/dnsdb/dnsdbq`, most DomainTools DNSDB API subscribers will just use `dnsdbq` to make interactive queries. For example, let's see use `dnsdbq` to see what IPv4 addresses have been used by `www.whitman.edu`:

```
$ dnsdbq -r www.whitman.edu/A
;; record times: 2014-03-26 21:27:00 .. 2022-04-18 17:13:39 (~8y ~24d)
;; count: 1616593; bailiwick: whitman.edu.
www.whitman.edu.  A  199.89.174.11
;; record times: 2010-06-24 13:49:41 .. 2014-03-27 16:40:25 (~3y
~277d)
;; count: 940920; bailiwick: whitman.edu.
www.whitman.edu.  A  199.89.174.13
;; record times: 2022-04-18 17:23:31 .. 2022-12-22 21:52:41 (~248d 4h
29m)
;; count: 117102; bailiwick: whitman.edu.
www.whitman.edu.  A  216.176.184.235
```

The DNSDB output shown above is in "presentation format" (which is meant to be easily readable by human beings). Many subscribers who are professional data analysts may prefer dnsdbq's JSON Lines output mode, perhaps "pretty printing" their results using `jq` (see `https://stedolan.github.io/jq/`):

```
$ dnsdbq -r www.whitman.edu/A -j -T datefix -S -k last -A7d | jq '.'
{
  "count": 117102,
```

```
  "time_first": "2022-04-18 17:23:31",
  "time_last": "2022-12-22 21:52:41",
  "rrname": "www.whitman.edu.",
  "rrtype": "A",
  "bailiwick": "whitman.edu.",
  "rdata": [
    "216.176.184.235"
  ]
}
```

`dnsdbq` has many options which can be used to control DNSDB output format. Decoding the ones used above:

- **`-r`** means we're going to search Rrnames (the "left hand side") of DNS resource records in DNSDB
- **`www.whitman.edu/A`** means "find 'A' records for that fully qualified domain name (FQDN)
- **`-j`** asks for output to be provided in JSON Lines format (see `https://jsonlines.org/`)
- **`-T datefix`** means we want human-readable values, not Un*x ticks (`https://www.unixtimestamp.com/`)
- **`-S`** means "sort results in descending order"
- **`-k`** last means "the sort key's the time each RRset was last seen"
- **`-A7d`** means "only return results seen in the last seven days"
- **`|`** ("pipe" or "vertical bar") takes the output from `dnsdbq` and sends it on to `jq`
- **`jq '.'`** means "prettyprint the JSON input"

You can read more about these and other dnsdbq options using:

```
$ man dnsdbq
```

## (2) Making Multiple DNSDB RRtype ="ANY" Queries Using dnsdbq Queries in Serial Batch Mode

Assume we want to make a bunch of DNSDB queries, such as queries for a set of 7,432 dot edu domain names (see Misc-App-A) for the arbitrary month-long period running from:

```
(Unix seconds) 1668377911        Sun, November 13, 2022 10:18:31 PM
UCT
(Unix seconds) 1670969911        Tue, December 13, 2022 10:18:31 PM
UCT
```

To use `dnsdbq` in batch mode, we begin by preparing a list of the commands we want to run, as described in the `dnsdbq` man page. Our queries will all be left hand wildcard searches, so that DNSDB will match all fully qualified domain names (FQDNs) using the specified base domains. Because we've not specified a specific RRtype, `dnsdbq` will default to returning results for all non-DNSSEC RRtypes (e.g., "A" records, "CNAME" records, "AAAA" records, "NS" records, "MX" records, "TXT" records, "SOA" records, etc.):

```
$ cat edus-dnsdbq-batch-job.txt
rrset/name/*.22cf.edu
rrset/name/*.290tech.edu
rrset/name/*.3ponts.edu
[…]
rrset/name/*.zoni.edu
rrset/name/*.zuvbpcqanl.edu
```

We can then run that batch of commands through `dnsdbq` with the **`-f`** (batch mode) option:

```
$ date ; dnsdbq -T datefix -j -l0 -A "1668377911" -B "1670969911" -f <
edus-dnsdbq-batch-job.txt >
edus-dnsdbq-batch-job-output-sequential.jsonl ; date
Tue Dec 20 13:50:36 PST 2022
dnsdbq: batch line status: NOERROR (no results found for query.)
[etc]
dnsdbq: batch line status: NOERROR (no results found for query.)
Tue Dec 20 14:44:18 PST 2022
```

In this run, as in the others that follow, we consider two fundamental questions:

***Fundamental Question 1:*** *How Long Did It Take For Our Queries to Run?* It took (14:44:18 - 13:50:36) => 53 min 42 sec (or 3,222 seconds) to run all 7,432 DNSDB queries sequentially. Obviously other queries, made from other workstations over different network connectivity will have different absolute performance – what matters here is relative performance.

***Fundamental Question 2:*** *How Many Results Did We Receive?* Our results consist of 32,844,287 lines. We track those query results to ensure results are (relatively) consistent, run-to-run, in this testing. Different queries would obviously return different result counts.

```
$ wc -l edus-dnsdbq-batch-job-output-sequential.jsonl
32,844,287 edus-dnsdbq-batch-job-output-sequential.jsonl     (commas
added manually for readability)
```

Our results include 7,432 "--" separator lines, one separator line (inserted by `dnsdbq`) between each set of results.

Deleting those separator lines leaves us with **32,836,855 results**. Those results look like:

```
$ more edus-dnsdbq-batch-job-output-sequential.jsonl
{"count":545,"time_first":"2018-03-12 07:00:02","time_last":"2022-12-19
08:00:02","rrname":"22cf.edu.","rrtype":"A","bailiwick":"22cf.edu.","rd
ata":["91.195.240.103"]}
{"count":60119,"time_first":"2010-07-05
13:15:44","time_last":"2022-12-19
08:00:02","rrname":"22cf.edu.","rrtype":"NS","bailiwick":"edu.","rdata"
:["dns1.name-services.com.","dns2.name-services.com.","dns3.name-servic
es.com.","dns4.name-services.com."]}
{"count":16583,"time_first":"2010-07-05
13:15:44","time_last":"2022-11-29
17:30:36","rrname":"22cf.edu.","rrtype":"NS","bailiwick":"22cf.edu.","r
data":["dns1.name-services.com.","dns2.name-services.com.","dns3.name-s
ervices.com.","dns4.name-services.com.","dns5.name-services.com."]}
[…]
{"count":33,"time_first":"2021-11-03 17:57:31","time_last":"2022-12-17
02:39:06","rrname":"testservices.zoni.edu.","rrtype":"A","bailiwick":"z
oni.edu.","rdata":["3.88.99.89"]}
```

## (3) Repeating Our dnsdbq RRtype ="ANY" Queries in dnsdbq Parallel Batch Mode

`dnsdbq` also has a **parallel** "batch" mode that you can enable with the **`-fm`** option. Can dnsdbq parallel batch mode "beat" the speed of the dnsdbq serial batch mode? To check, we'll rerun our command file of entries with:

```
$ date ; dnsdbq -T datefix -j -l0 -A "1668377911" -B "1670969911" -fm <
edus-dnsdbq-batch-job.txt > edus-dnsdbq-batch-job-output-parallel.jsonl
; date
 Wed Dec 28 10:27:43 PST 2022
 Wed Dec 28 10:33:39 PST 2022
```

The change between that command and the serial batch mode command is obviously tiny, **just one letter.**

It took just (10:33:39 - 10:27:43) = 5 minutes 56 seconds (aka 356 seconds) to run the same set of queries in parallel batch mode (using up to 10 concurrent streams). That's much better! We received 33,222,783 results from the parallel mode batch queries, slightly more than we received for our serial

mode batch queries, so this speedup is obviously not at the expense of some results previously received.

```
$ wc -l edus-dnsdbq-batch-job-output-parallel.jsonl
33,222,783 edus-dnsdbq-batch-job-output-parallel.jsonl    (commas
added manually)
```

**The big difference between our serial dnsdbq queries and our parallel dnsdbq queries is the difference in speed achieved:**

| | Elapsed time | Relative Speedup to serial batch mode | Returned results |
|---|---|---|---|
| **dnsdbq serial batch mode** | 3,222 sec | 1.0 (baseline) | 32,836,855 |
| **dnsdbq parallel batch mode** | **356 sec** | **9.05** | **33,222,783** |

Obviously, the parallel batch approach offers a **HUGE speedup!** Given that this is an embarrassingly parallel problem, we might expect parallel execution to deliver up to 10X the performance seen for sequential execution, but our workload may experience network contention, I/O contention, limitations due to finite memory, or suffer from other impacts that result in less-than-perfect parallel speedup. Still, parallel execution running over 9X as fast as sequential execution is "nothing to sneeze at."

Let's now move on to working with DNSDB API from Python3.

# PART II: Sequential DNSDB API Execution via Python3

(https://www.domaintools.com/resources/user-guides/farsight-dnsdb-api-version-2-documentation/)

> ***"Doing one thing at a time is the essence of Zen."***

The runs in the preceding section were all done using `dnsdbq`. Let's now look at what a developer (such as a data scientist or application programmer) might do to run DNSDB query code serially via Python3.

- *Single Minimal Query:* We'll begin by making a single query with DNSDB API from Python3, just to get a sense of what's involved in making ANY DNSDB API query from Python3.
- *Sequential Python3 Code:* We'll then create sample code that makes a batch of DNSDB API queries sequentially from Python3 as a throughput baseline. This will also let us introduce some Python3 utility functions we'll need for both this baseline program and subsequent iterations.

## (4) Making A Single Simple DNSDB API Query from Python3

At its most basic, it's relatively easy to call DNSDB from Python3. Let's look at a minimal program designed to make an "A" record query for `www.whitman.edu/A` using the `requests` http client library (https://requests.readthedocs.io/en/latest/):

```
$ cat single_query.py
#!/usr/local/bin/python3
""" run a single DNSDB API query """
# https://requests.readthedocs.io/en/latest/
import requests
key = "insertYourRealAPIkeyHere"
myfqdn = "www.whitman.edu/A"
url = "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/" + myfqdn
myheaders = {'X-API-Key': key, 'Accept': 'application/jsonl'}
r = requests.get(url, headers=myheaders, timeout=60)
print(r.content.decode())
```

Decoding our little program:

```
#!/usr/local/bin/python3
```

The above line points at the Python3 interpreter we want to use to run our code.

```
""" run a single DNSDB API query """
```

This triple double-quoted line is a “docstring” and helps to document the purpose of each block of code.

```
# https://requests.readthedocs.io/en/latest/
import requests
```

We will use the `"requests"` library to call DNSDB API; because this is a 3rd-party library, we must formally import it. We include a comment pointing at the 3rd-party library's home page (for our reference if we need details about this library and its calls).

```
key = "insertYourRealAPIkeyHere"
```

To access DNSDB, we need to supply our DNSDB API key – of course, actually hardcoding a credential in a program isn't a very secure or scalable approach, but doing so helps keep this example simple.

```
myfqdn = "www.whitman.edu/A"
```

This is the query we want to make – we're going to ask for "A" records for the domain `"whitman.edu"`

```
url = "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/" + myfqdn
```

Making a DNSDB API call requires us to know the format for that API call. It is described in `https://www.domaintools.com/resources/user-guides/farsight-dnsdb-api-version-2-documentation/`

```
myheaders = {'X-API-Key': key, 'Accept': 'application/jsonl'}
```

Our DNSDB API call requires a special header passing the API key, and declaring the format results we want.

```
r = requests.get(url, headers=myheaders, timeout=60)
```

We include a 60 second timeout just in case something goes "crazy" (normally this call will only take few seconds).

```
print(r.content.decode())
```

We're printing out our results. `"r.content"` returns the results of the call; `decode()` changes bytes received to a string.

That's all we need to call DNSDB API in Python to process a single query. It isn't particularly elegant, but it's succinct and it works. A full copy of this code can be found in Python3-App-A. Let's now see what the output from that program looks like:

```
$ ./single_query.py
{"cond":"begin"}
{"obj":{"count":1616593,"time_first":1395869220,"time_last":1650302019
,"rrname":"www.whitman.edu.","rrtype":"A","bailiwick":"whitman.edu.","
rdata":["199.89.174.11"]}}
{"obj":{"count":940920,"time_first":1277387381,"time_last":1395938425,
"rrname":"www.whitman.edu.","rrtype":"A","bailiwick":"whitman.edu.","r
data":["199.89.174.13"]}}
{"obj":{"count":120307,"time_first":1650302611,"time_last":1672839402,
"rrname":"www.whitman.edu.","rrtype":"A","bailiwick":"whitman.edu.","r
data":["216.176.184.235"]}}
{"cond":"succeeded"}
```

This code and its output have numerous characteristics worth noting:

- We hardcoded our API key in the program – that's not a best practice from a security (or code maintenance) point of view.
- We also hardcoded the domain name and record type we wanted to retrieve – something a *little* more flexible would help generalize this code (you don't want to have to edit the program everytime you want to run a different query!)
- We implicitly assume that everything will run without any problems, so we don't check for (nor handle) any potential errors. That's probably unduly optimistic.
- Our output is in raw JSON Lines format rather than a nicely-formatted report. You can train yourself to read JSON Lines output, but you shouldn't have to.
- Times are shown in Un*x ticks (see `https://www.unixtimestamp.com/`) rather than a more "human-readable" format.
- Our three actual results are bracketed by `'{"cond":"begin"}'` and `'{"cond":"succeeded"}'` – those lines are a new part of the enhanced DNSDB API Version 2 "Streaming API Framing" ("SAF") Protocol (see `https://www.domaintools.com/resources/user-guides/farsight-streaming-`

`api-framing-protocol-documentation/)`. We probably don't need to routinely see the Streaming API Framing Protocol messages printed out.
- If we were to interrupt execution of our program by hitting a control-C, that key sequence would interrupt it, but it would show us a stack dump by default – that's not a very "polished" user experience.
- We also didn't specify how many results to accept (so we're limited to no more than 10,000 by default), nor did we specify a time fence (so we should expect to see results that may go back all the way to June 2010).

A major portion of our eventual "production" serial code is devoted to cleaning up the above issues.

# (5) "More Professional and Usable" DNSDB API Serial Query Code

Illustrative as our simple single-query program might have been, let's build something a little more professional (and usable!) next.

## (5.1) Getting Our API Key From ~/.dnsdb-apikey.txt

Rather than hardcoding an API key in our code, we'll retrieve the API key from a dot file in our home directory at run time. We'll handle this task as a separate function defined near the top of our file. In this case, no arguments were passed into the function:

```
def get_api_key():
```

You may also notice that in this function, as in all of our other routines, we've included a "doc string" in triple double quotes. It is like a comment, describing what the routine does. If you create a function without one, `pylint` will complain (and we don't want that):

```
    """ Retrieve the DNSDB API key """
```

Now we'll construct the file path by:

- Retrieving the home directory path with `Path.home`
- Using the `os.path.join` function to combine the home directory path and the file name. The `os.path.join` function will automatically employ an OS-appropriate path separator, e.g., "/" in the case of MacOS and other Un*x-related operating systems.

```
 api_key_file_path = os.path.join(str(Path.home()),
".dnsdb-apikey.txt")
```

With the file name established, we're ready to try opening the file and retrieving the API key from it. Sometimes the API key file may not exist – if that's the case, we should return an error and exit. We'll handle that by using a "`try/except`" structure. If all goes well, we'll return the API key value to the caller.

```
        try:
            with open(api_key_file_path, encoding='UTF-8') as my_api_file:
                val = my_api_file.readline().strip()
        except FileNotFoundError:
            exit("DNSDB API key should be in ~/.dnsdb-apikey.txt")
        return val
```

## (5.2) Handling Control-C Interrupts Cleanly

If we don't have a handler defined, hitting control-C to interrupt execution of our code reacts pretty brutally, resulting in just a raw stack dump as a response to the interrupt. We'll improve on that default approach by defining a handler for the interrupt:

```
    def handler(_ , _2):
        """Stub routine to handle the user hitting ctrl-C"""
        exit("\nUser hit ctrl-C -- exiting")
```

Since we don't need to do anything with the arguments passed to the handler, we'll just define those "arguments" as stub arguments (`"_" and "_2"`). We'll actually instantiate the handler in our main program.

## (5.3) Printing Our JSON Results in an Easy-to-Read Columnular Format

While it isn't essential for us to reformat our output into an easy-to-read format, it's a pretty basic step. Our results arrive as JSON Lines objects. "Pretty printing" one of those, it looks like:


```
{
  "obj": {
    "count": 1616593,
    "time_first": 1395869220,
    "time_last": 1650302019,
    "rrname": "www.whitman.edu.",
    "rrtype": "A",
    "bailiwick": "whitman.edu.",
    "rdata": [
      "199.89.174.11"
    ]
```

```
    }
}
```

We begin by loading a result into a Python3 JSON formated object, after which we'll be able to easily extract specific elements of it:

```
def print_detailed_bits(myrecord):
    """format results for display"""
    parser = cysimdjson.JSONParser()
    myrecord_json_format = parser.loads(myrecord)
```

Many might be tempted to just use the stock JSON library to parse our results, but numerous third party libraries claim to be faster, including (in alphabetical order):

- ***cysimdjson*** (https://github.com/TeskaLabs/cysimdjson and https://github.com/simdjson/simdjson) "Fast JSON parsing library for Python, 7-12 times faster than standard Python JSON parser."
- ***msgspec*** (see https://github.com/jcrist/msgspec) For supported types, encoding/decoding a message with msgspec can be ~2-40x faster than alternative libraries."
- ***orjson*** (see https://github.com/ijl/orjson) "orjson is a fast, correct JSON library for Python. It benchmarks as the fastest Python library for JSON […]"
- ***Python-rapidjson*** (see https://github.com/python-rapidjson/python-rapidjson) "RapidJSON is an extremely fast C++ JSON parser and serialization library […]"
- ***sajson*** (see https://github.com/chadaustin/sajson) "sajson's performance is excellent - it frequently benchmarks faster than RapidJSON, for example."
- ***ujson*** (see https://github.com/ultrajson/ultrajson) "UltraJSON is an ultra fast JSON encoder and decoder written in pure C with bindings for Python 3.7+."
- ***yyjson*** (see https://github.com/ibireme/yyjson) "Fast: can read or write gigabytes per second JSON data on modern CPU."

Some formal benchmarks for various Python JSON libraries can be seen at https://github.com/tktech/json_benchmark

We elected to go with `parser = cysimdjson.JSONParser()`
`myrecord_json_format = parser.loads(myrecord)`

Once we've loaded the JSON object, we can extract the resource record type by asking for the `'rrtype'` element from under the `'obj'` element:

```
my_rrtype = myrecord_json_format['obj']['rrtype']
```

We can then do things like use our newly-available Rrtype value to do things like keep just specific records, perhaps only "A" records and "CNAME" records.

```
# assume we're only interested in "A" records and CNAMEs
if my_rrtype.rstrip() not in ("A", "CNAME"):
    return None
```

Now let's extract a few other fields from our JSON format record:

```
my_rrname = myrecord_json_format['obj']['rrname']
my_count = myrecord_json_format['obj']['count']
my_rdata = str((myrecord_json_format['obj']['rdata']).export())
```

Dates and times are a little more complicated to work with. We can represent them in a variety of different "human readable" formats (see the *"strftime() and strptime() Format Codes"* section of `https://docs.python.org/3/library/datetime.html`, but we'll use *two-digit year - two-digit month - two-digit date* format followed by a space and then *two-digit hours : two-digit minutes*

```
myformat = '%y-%m-%d %H:%M'
```

Formatting date times in DNSDB is complicated by the two different types of results we may receive: results from *sensor* data, or results from *"czds"/"zone file"* data – these two data sources have different date time field names. We also need to handle both the time **first** seen, and the time **last** seen, both of which get reported. Our code to handle all that looks like:

```
# time_last
try:
    extract_tl = myrecord_json_format['obj']['time_last']
except KeyError:
    extract_tl = myrecord_json_format['obj']['zone_time_last']
enddatetime = strftime(myformat, gmtime(extract_tl))
# time_first
try:
    extract_tf = myrecord_json_format['obj']['time_first']
except KeyError:
    extract_tf = myrecord_json_format['obj']['zone_time_first']
startdatetime = strftime(myformat, gmtime(extract_tf))
```

We could also simply have elected to take advantage of "human format" date time strings from DNSDB API Version 2, thereby eliminating our need to use strftime. See the DNSDB API Version 2 API reference documentation for more details on this.

We still need to decide how we want to format some of our other variables for display. Python3.6 and later can use "f strings" for formatting (see `https://docs.python.org/3/tutorial/inputoutput.html`). We'll need to decide how much space to devote to each field. For example, most RRnames are fairly short, but some RRnames may be quite long. As a matter of judgment, we choose to allow up to 55 characters (left justified) for this field (any longer RRnames will push everything else on the line over to the right).
The RRtype field gets six characters of space (left justified) to ensure there's room for longish RRtype names such as "CNAME".

The count field is alloted 12 spaces, right justified, with comma separators to help make large values more easily readable:

```
        # format the output for display
        my_rrname = f'{my_rrname:<55}'
        my_rrtype = f'{my_rrtype:<6}'
        my_count = f'{my_count:>12,}'
```

Finally, we'll use another f string to "paste the various elements together" into a single string to return. Because we want to treat both of our two date-times as single fields, we'll double quote them using backslash-escaped double quote marks.

```
        # quoting the date times requires that we use escaped quotes
        results = f'{my_rrname} {my_rrtype} \"{enddatetime}\"
\"{startdatetime}\" {my_count} {my_rdata}'
        return results
```

## (5.4) Stripping Streaming API Framing (SAF) Records

Steaming API Framing (SAF) records are designed to alert developers and users to potentially incomplete results (perhaps caused by server-side timeouts or server-side result caps). While SAF records can be carefully scrutinized and acted upon, in this example we're just going to strip and discard them. We leverage the fact that the results (including the SAF records) are stored in a list, allowing us to use stack manipulation operations such as `pop(0)` to pop the first list item, and `pop()` to pop the last list item (after we've verified that they ARE indeed SAF records!)

```
    def remove_saf_entries(mylist):
        """ Remove the Streaming API Framing Records From the Results """
        mylist2 = mylist.splitlines()
        # Strip the streaming format bookend records
        if mylist2[0] == '{"cond":"begin"}':
            mylist2.pop(0)
        if ((mylist2[-1] == '{"cond":"succeeded"}') or \
```

```
        (mylist2[-1] == '{"cond":"limited","msg":"Result limit
reached"}')):
        mylist2.pop()
    return mylist2
```

We're now ready to handle making the actual DNSDB query. That routine begins simply

```
def make_query(myfqdn):
    """ Make the DNSDB query for myfqdn """
```

but immediately raises some questions, such as…

## (5.5) WHERE Are We Going to Send Our Queries?

Most DNSDB API users will make their queries to `api.dnsdb.info`. However, if you resolve that name, you'll see that it actually points at TWO different IPs:

```
$ dig api.dnsdb.info
[…]
api.dnsdb.info.         3600  IN    CNAME dnsdb.info.
dnsdb.info.       3600  IN    A     104.244.14.69
dnsdb.info.       3600  IN    A     104.244.13.65
```

Those two IPs point at geographically distributed datacenters located in Dulles, Virginia and Palo Alto, California. Checking both from the author's location in Oregon, there's a factor of two difference in latency:

```
$ ping -c 15 api-pao.dnsdb.info
PING api-pao.dnsdb.info (104.244.13.65): 56 data bytes
64 bytes from 104.244.13.65: icmp_seq=0 ttl=47 time=37.798 ms
[…]

--- api-pao.dnsdb.info ping statistics ---
15 packets transmitted, 15 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 36.425/43.396/52.659/4.507 ms

$ ping -c 15 api-iad.dnsdb.info
PING api-iad.dnsdb.info (104.244.14.69): 56 data bytes
64 bytes from 104.244.14.69: icmp_seq=0 ttl=38 time=81.539 ms
[…]

--- api-iad.dnsdb.info ping statistics ---
15 packets transmitted, 15 packets received, 0.0% packet loss
```

```
round-trip min/avg/max/stddev = 81.539/88.876/98.124/4.352 ms
```

Latency is important both for "chatty" protocols (which require multiple round trips to negotiate parameters), and for bulk TCP flows where the *bandwidth-delay product* limits throughput (https://en.wikipedia.org/wiki/Bandwidth-delay_product).
We're going to force our API endpoint to use the closer-to-the-author DNSDB API server in Palo Alto, CA, `api-pao.dnsdb.info`. This will minimize our network latency, and also ensure that the API doesn't potentially flop mid-test from one DNSDB API server to a completely different one. [For those who may be on the East Coast, `api-iad.dnsdb.info` in Dulles, VA may be closer].

**Caution:** Hard-coding a specific DNSDB API endpoint is NOT something that should be done "casually" – on rare occasions, one server location or another may be offline for maintenance. This is normally "operationally transparent" – that is, ***IF*** you simply use the generic `api.dnsdb.info` endpoint. If you hardcode a *specific* DNSDB API endpoint (without telling your code to fall back to the other server node as a backup if necessary), you might find yourself actually experiencing an apparent service outage (even if it's really just a single DNSDB API node that's offline for routine maintenance).

**Also Note:** DNSDB API endpoints are accessible over **both IPv4 and IPv6,** but the network path over those two protocols may be different. For example, traffic over IPv4 connectivity may use one network service provider and traffic over IPv6 connectivity may use a different provider. This may (or may not) impact performance and ultimately your choice of endpoint. In the author's case, use of IPv6 would NOT change his preference for `api-pao.dnsdb.info` over `api-iad.dnsdb.info`:

```
$ ping6 -c 15 api-pao.dnsdb.info
PING6(56=40+8+8 bytes) […] 2620:11c:f004::65
16 bytes from 2620:11c:f004::65, icmp_seq=0 hlim=47 time=130.601 ms
[…]

--- api-pao.dnsdb.info ping6 statistics ---
15 packets transmitted, 15 packets received, 0.0% packet loss
round-trip min/avg/max/std-dev = 35.250/64.543/155.264/41.741 ms
$ ping6 -c 15 api-iad.dnsdb.info
PING6(56=40+8+8 bytes) […] 2620:11c:f008::69
16 bytes from 2620:11c:f008::69, icmp_seq=0 hlim=39 time=156.779 ms
[…]

--- api-iad.dnsdb.info ping6 statistics ---
15 packets transmitted, 15 packets received, 0.0% packet loss
round-trip min/avg/max/std-dev = 88.891/110.719/248.337/40.726 ms
```

## (5.6) WHAT Will We Use for Our Client HTTP Library? And Will We Use Sessions?

We decided to make our call to the DNSDB API with the `requests` library. Some may wonder "Why use the `requests` library for this rather than some other alternative http client library?" This is a valid question, and numerous alternatives DO exist. We chose `requests` because it's fast, easy to use and close to being a *de facto* standard at this point. Nonetheless, if you want to explore alternatives, a few popular options to consider include (in alphabetical order):

- ***Aiohttp*** (https://docs.aiohttp.org/en/stable/) – we'll use this for our `asyncio` testing later in this document
- ***aiosonic*** – reportedly quite fast
- ***Httpx*** *(https://www.python-httpx.org/)* has support for HTTP/2
- ***Pycurl*** (http://pycurl.io/) – a long time favorite
- ***`Pycurl-requests`*** (https://pypi.org/project/pycurl-requests/) `()` – a requests-compatible interface to pycurl
- ***`Urllib3`*** (https://urllib3.readthedocs.io/en/stable/) `(` this is the library "underneath" requests

We also explicitly decide to NOT use the `request's sessions` functionality given that `request's sessions` functionality (see `https://requests.readthedocs.io/en/latest/user/advanced/)` is believed to NOT be thread safe (see `https://github.com/psf/requests/issues/1871` for a discussion of this). If you believe use of `sessions` IS thread-safe, enabling use of `sessions` may result in a **MATERIAL** improvement in run times. For example, sneaking a peak at some code from later in this document:

```
$ ./run_queries_threaded_mode.py < all-edus.txt >
threaded_without_session.txt
Elapsed time:       452.950 seconds
$ ./run_queries_threaded_mode_with_session.py < all-edus.txt >
threaded_with_session.txt
Elapsed time:       319.977 seconds
```

The only difference between the two codes (differences manually highlighted in the diff below):

```
$ diff run_queries_threaded_mode.py
run_queries_threaded_mode_with_session.py
22a23,25
> # enable Requests session
> s = requests.Session()
>
48c51
<       r = requests.get(url, headers=myheaders, timeout=3600)
```

```
---
>       r = s.get(url, headers=myheaders, timeout=3600)
```

## (5.7) Handle Making Our Actual DNSDB Query

Let's now come back to building the URL we'll pass for our API call…Obviously, we'll need to pass the domain we want to lookup as part of that URL. We'll also ask for up to a million results per query, and set the time fencing window to the same dates we used in our `dnsdbq` batch queries. We'll also set up the header we need to send with our API key, and pass the required `"Accept"` value:

```
    url = "https://api-pao.dnsdb.info/dnsdb/v2/lookup/rrset/name/" +
myfqdn + \

"?limit=1000000&time_last_after=1668377911&time_first_before=167096991
1"
    myheaders = {'X-API-Key': my_apikey, 'Accept':
'application/jsonl'}
```

In generalizable production code, we would NOT want to hardcode `limit=1000000` (nor specific time fences!), but for our testing purposes, those fixed settings make sense.

*Optional:* If you want to have your client program (and its version) show up in DNSDB API usage summaries, you can add two parameters to the above URL:

```
&swclient=myappname&version=0.3
```

Both of those fields should be customized for your code, and each string be limited to no more than twenty characters each.

Now we're ready to make our call to DNSDB API and handle the results (if everything went well), including processing our results. "Processing our results" in this case means stripping the SAF entries, arbitrarily keeping only "A" records and "CNAME" records, formatting the output for display, and printing the results to STDOUT. (If something did go wrong and we receive a **NON**-200 status code, we'll log that error to STDERR, instead.) Just to minimize any potential problems, we'll set our timeout to 3600 seconds (1 hour), far longer than the server-side timeout (DNSDB API has a hard-coded timeout of 15 minutes).

```
    r = requests.get(url, headers=myheaders, timeout=3600)
    # Status Code 200 == Success
    if r.status_code == 200:
        stripped_results = remove_saf_entries(r.text)
        for myfqdns in stripped_results:
            myline = print_detailed_bits(myfqdns)
            if myline is not None:
                print(myline)
    else:
```

```
        sys.stderr.write(myfqdn+" returned status
code="+r.status_code+"\n")
```

## (5.8) Importing Our Required Libraries

We'll need `import` lines to give us access to the library routines we're relying on. These `import` lines go at the top of our program. We generally list the libraries in alphabetical order, with Python3-internal libraries first and 3rd-party libraries (such as`"cysimdjson"` and `"requests"`) following in alphabetical order. We also like to provide a pointer to documentation for any 3rd-party libraries. This can be helpful for people who may not already have these libraries pre-installed.

```
#!/usr/local/bin/python3
""" run_queries.py """
import os
from pathlib import Path
from signal import signal, SIGINT
import sys
# pylint: disable=W0622
from sys import exit, stdin
import time
from time import strftime, gmtime
# https://github.com/TeskaLabs/cysimdjson
import cysimdjson
# https://requests.readthedocs.io/en/latest/
import requests
```

Keen-eyed readers may also note a `"# pylint: disable=W0622"` comment hidden amongst our `import` lines. That command tells `pylint` to disregard the fact that we're overriding the built-in `exit` command. We know we're doing it, we want to do it, and it's OK for us to do so. That special comment suppresses the warning that `pylint` would otherwise generate when checking this program. We'll supress some other pylint cosmetic complaints from time-to-time later in this document, too. We also will use the `radon` utility (see `https://pypi.org/project/radon/)` to ensure the Python3 code we write is maintainable and comprehensible.

## (5.9) The Main Routine

All that's left now is handling the main routine where our program begins to run when we execute it. The first thing we're going to do is make sure we've got a stream of domains coming in from STDIN for us to look up:

```
if __name__ == "__main__":
```

```
# Did you forget to pipe in the sites to check?
if stdin.isatty():
    exit("Pipe in sites to check via stdin")
```

We're now ready to get our elapsed time for timing, to instantiate our control-C interrupt handler, and to get our DNSDB API key:

```
# get timing start time
start_time = time.time()
# handle ctrl-C interrupt cleanly
signal(SIGINT, handler)
# get the DNSDB API key
my_apikey = get_api_key()
```

We're now ready to read the domains we want to process. We're going to work domain-by-domain, reading a domain and then immediately processing it, looping until we're out of domains. We strip any trailing newlines as part of that ingestion process:

```
for line in stdin:
    make_query(line.strip())
```

We'll then close out our timing loop:

```
# report elapsed time
elapsed_time = time.time() - start_time
elapsed_time = f'{elapsed_time:>12.3f} seconds\n'
sys.stderr.write("Elapsed time: " + elapsed_time)
```

A full copy of the full program can be found in Python3-App-B. We'll test this code with our file of 7,432 dot edu domain names (copy of those domains in Misc-App-A).
Running that program, we see:

```
$ ./run_queries_serial.py < all-edus.txt > all-edus-results-serial.txt
Elapsed time:     4067.185 seconds
$ wc -l all-edus-results-serial.txt
20,814,548 all-edus-results-serial.txt [commas added manually for readability]
```

If you wonder, "Why were there 32 million results when we ran `dnsdbq`, but 'only' about 20.81 million results here?", remember that our Python3 sequential code is intentionally keeping only "A" and "CNAME" records, while our earlier `dnsdbq` runs included ALL non-DNSSEC record types. You can see this by browsing the sample output from that file:

```
arizona.edu.                          A      "23-01-04 20:18" "10-06-24 03:07"   83,170,060 ['128.196.128.233']
b.arizona.edu.                        CNAME  "23-01-03 15:27" "11-06-21 21:03"        3,920 ['view.medu.com.']
m.arizona.edu.                        CNAME  "23-01-04 09:39" "14-08-20 17:54"      529,071 ['uarizona-prodweb.modolabs.net.']
www.m.arizona.edu.                    CNAME  "22-11-19 06:24" "16-07-03 13:26"           69 ['uarizona-prodweb.modolabs.net.']
u.arizona.edu.                        A      "23-01-04 13:55" "15-08-05 13:29"      381,555 ['128.196.130.121']
www.u.arizona.edu.                    CNAME  "23-01-04 17:17" "15-08-05 12:21"    1,067,662 ['sage.u.arizona.edu.']
menu.u.arizona.edu.                   CNAME  "22-12-31 16:42" "15-10-16 06:43"          298 ['sage.u.arizona.edu.']
sage.u.arizona.edu.                   A      "23-01-04 12:16" "15-08-05 12:21"    1,479,692 ['128.196.130.121']
shell.u.arizona.edu.                  CNAME  "22-12-29 08:06" "15-08-26 20:55"          222 ['sage.u.arizona.edu.']
koshersalt.u.arizona.edu.             A      "22-12-28 18:51" "16-12-07 17:42"        1,146 ['128.196.130.19']
22.arizona.edu.                       CNAME  "23-01-03 09:21" "17-12-21 23:39"        3,506 ['live-uofapres.pantheonsite.io.']
af.arizona.edu.                       A      "23-01-03 18:20" "10-07-27 20:47"        5,616 ['128.196.132.134']
www.af.arizona.edu.                   A      "22-11-24 18:10" "10-08-28 19:54"           28 ['128.196.132.134']
ag.arizona.edu.                       A      "23-01-04 08:46" "21-11-16 22:17"       93,889 ['44.235.112.45']
www.ag.arizona.edu.                   CNAME  "23-01-04 19:22" "16-04-06 22:59"      130,284 ['cals.arizona.edu.']
erl-donb.erlab.ag.arizona.edu.        A      "23-01-04 08:42" "16-12-14 19:25"        1,159 ['150.135.106.14']
erl-jwood.erlab.ag.arizona.edu.       A      "23-01-04 08:42" "16-11-30 00:44"        1,181 ['150.135.106.3']
erl-kbell.erlab.ag.arizona.edu.       A      "22-11-28 12:57" "16-12-22 04:20"        1,179 ['150.135.106.13']
erl-kfitz.erlab.ag.arizona.edu.       A      "22-11-26 07:38" "11-07-03 03:39"        1,096 ['150.135.106.10']
erl-dmoore.erlab.ag.arizona.edu.      A      "23-01-03 09:25" "16-12-27 18:16"        1,102 ['150.135.106.6']
erl-eglenn.erlab.ag.arizona.edu.      A      "23-01-04 08:42" "16-10-24 19:49"        1,154 ['150.135.106.9']
erl-jriley.erlab.ag.arizona.edu.      A      "22-12-31 02:57" "16-10-29 08:34"        1,150 ['150.135.106.11']
erl-server.erlab.ag.arizona.edu.      A      "22-12-30 22:57" "10-08-19 16:39"        6,400 ['150.135.106.2']
```

## (6) Is There Much Variation in Result Counts Over Multiple Runs?

While our primary focus is on "speeding things up," we want to ensure that doing so is not going to negatively impact the quantity of results returned. Let's repeat our query ten times and see what sort of variation we discover. (We'd love to run each of our test codes hundreds of times, but there are limits to what's practical and reasonable – we just want to give you a sense of what you might encounter). We'll do those runs using a small shell loop:

```
$ for i in {1..10}
> do
> ./run_queries_serial.py < all-edus.txt > serial-results.txt
> wc -l serial-results.txt
> done
Elapsed time:     4250.735 seconds
20815193 serial-results.txt
Elapsed time:     4555.443 seconds
20815631 serial-results.txt
Elapsed time:     4989.667 seconds
20816124 serial-results.txt
Elapsed time:     5112.302 seconds
20824176 serial-results.txt
Elapsed time:     4607.801 seconds
20824702 serial-results.txt
Elapsed time:     4429.962 seconds
20825376 serial-results.txt
Elapsed time:     3502.503 seconds
20825779 serial-results.txt
Elapsed time:     3306.125 seconds
20826240 serial-results.txt
Elapsed time:     3450.785 seconds
```

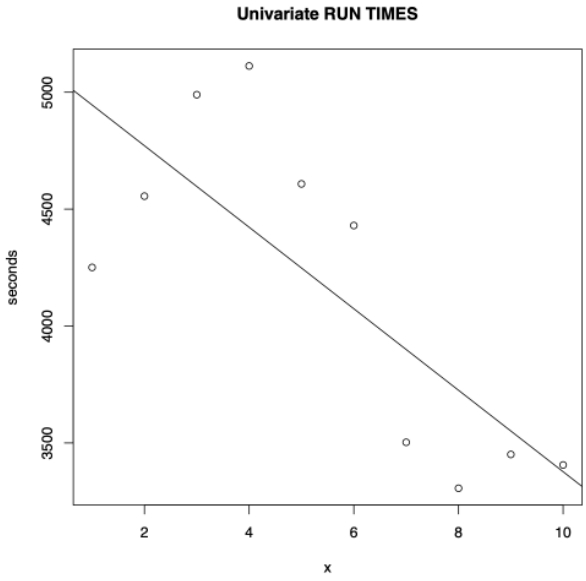
```
20826723 serial-results.txt
Elapsed time:     3405.594 seconds
20827005 serial-results.txt
```

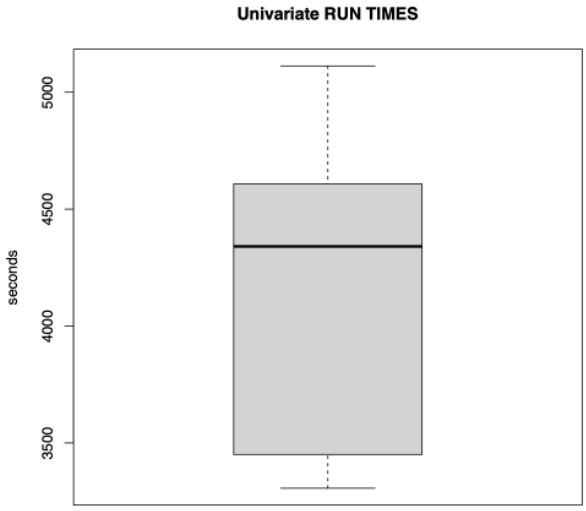
Clearly there ***IS*** some variation in both processing time and in the total results returned. It's hard to tell if there's a pattern to those results without visualizing them. Let's graph our results in "R," to see if we can get a better sense of what's going on (see `https://www.r-project.org/)`. "R" versions have often-somewhat-odd "nicknames" (derived from the animated comic strip "Peanuts" by Charles Schulz, see https://github.com/r-hub/rversions). In this case we're running R version 4.2.2, *aka* "Innocent and Trusting" (released 2022-10-31).

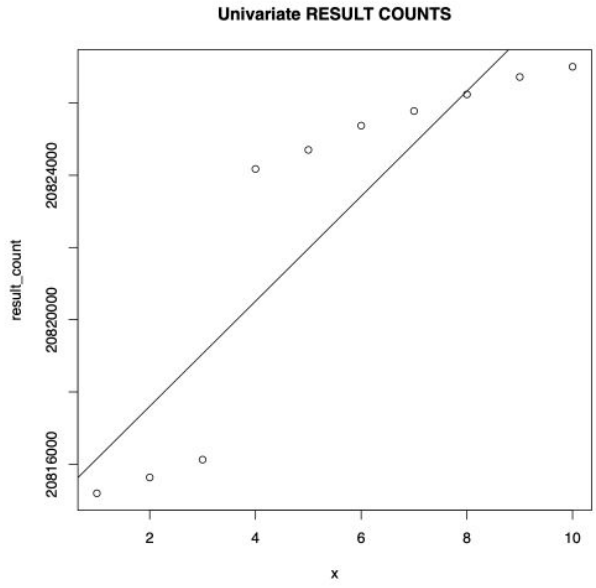
We've included a full copy of this code in R-App-A. Running that code, our univariate statistical results look like:

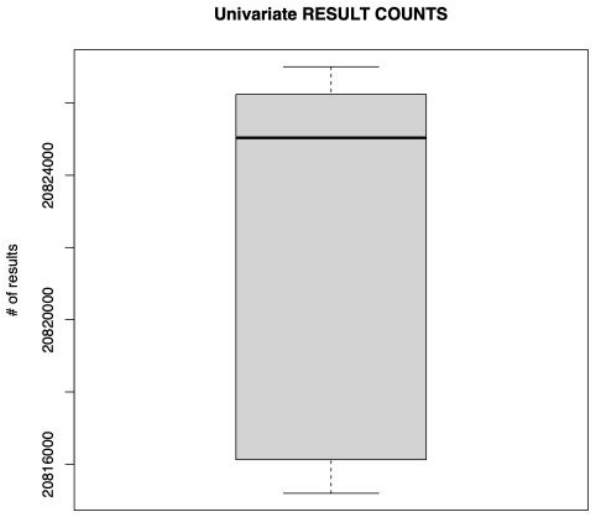
```
$ R -q < univariate-serial-python.R
[…]
> length(run_time)
[1] 10
> mean(run_time)
[1] 4161.092
> sd(run_time)
[1] 688.5368
> min(run_time)
[1] 3306.125
> median(run_time)
[1] 4340.349
> max(run_time)
[1] 5112.302
> max(run_time)-min(run_time)
[1] 1806.177
[…]
> length(result_count)
[1] 10
> mean(result_count)
[1] 20822695
> sd(result_count)
[1] 4940.156
> min(result_count)
[1] 20815193
> median(result_count)
[1] 20825039
> max(result_count)
[1] 20827005
```

```
> max(result_count)-min(result_count)
[1] 11812
[…]
```




It's hard to miss that the number of results (but not the run times) returned for successive runs monotonically increases. We believe that this is effectively the result of results being cached on the DNSDB API server. That's something that's beyond our ability to control (and represents a tiny level of variation in percentage terms). The good news is that we're not seeing any indication that results are NEGATIVELY impacted by repeated runs hitting the API.

# PART III: Parallel DNSDB API Execution via Python3

> ***"There should be one-- and preferably only one --obvious way to do it."***
> ***Tim Peters, "The Zen of Python"***
> ***(https://peps.python.org/pep-0020/)***

> ***"Some people, when confronted with a problem, think: 'I know, I'll use multithreading.' Nothhw tpe yawrve o oblems." (Eirikr Åsheim, 2012)***

> ***"Things I Wish They Told Me About Multiprocessing,"***
> ***(https://speakerdeck.com/pycon2019/pamela-mcanulty-things-i-wish-they-told-me-about-the-multiprocessing-module-in-python-3?slide=3)***

In this part we (finally!) get to the fun part of all this, running DNSDB API in parallel mode via Python3. We're going to test multiple approaches:

- ***Multiprocessing:*** Believe it or not, taking advantage of this powerful capability requires only relatively minor changes to our sequential code, and parallel multiprocessing delivers a **huge** improvement in throughput. As part of evaluating parallel multiprocessing, we'll look at the impact of using different pool access methods, too.
- ***Threading:*** Changing from parallel multiprocessing (potentially running on all available processors) to running with less-capable Python3 threads (running on just one processor servicing all the threads) is a matter of changing one library import line, so for completeness, we'll trial threading in this section as well.
- ***Asyncio:*** Then we're also going to show you how you can make "parallel" DNSDB API queries using asyncio in Python3. This can be as fast as our Python3 multiprocessing code, and is a more flexible alternative that some may prefer, although the overall complexity of asyncio code (and the potential opportunities for mis-steps) tend to be somewhat greater.

- ***Cythonized Asyncio:*** Some people worry that interpreted Python3 code is slow. We'll try Cythonizing our asyncio code to explore that possibility. Cython takes Python3 code as input, converting it into compilable C code.

## (7) Accessing DNSDB API with Python3 Parallel Mode

If you're using a workstation with multiple cores (and that's virtually all systems these days), just one of those cores will run typical Python3 code. Fortunately, Python3 also supports **multiprocessing.** Multiprocessing lets you take advantage of all the cores you may have available, running multiple jobs at the same time. In the "old days," making code parallel was a difficult and error-prone process, but these days you can readily make Python3 jobs take advantage of multiprocessing with little-to-no incremental effort.

**Put another way, most of our code from the `run_queries_serial.py` serial example in the previous section will go unchanged.**

The few differences between our old `run_queries_serial.py` and our new `run_queries.py` code are as follows:

- We need to add the multiprocessing library:

```
import multiprocessing as mp
```

- We also need to define the number of parallel processing streams we want to use:

```
# DNSDB API allows up to a maximum of 10 concurrent queries
processes=10
```

- In the `__main__` routine, we explicitly specify that we're creating new subprocesses with `spawn` (rather than `fork`). `spawn` is the default for macOS, but other Un*x operating systems may use `fork` by default, instead. That small detail can have big implications for variable inheritance (`https://docs.python.org/3/library/multiprocessing.html#contexts-and-start-methods`). We explicitly force use of `spawn` (rather than `fork`) in `__main__` as "insurance:"

```
# spawn is the default for MacOS, but fork is the default for many
other Un*x systems
# if spawn's not used, fork has implications for global variable
inheritance, etc.
mp.set_start_method('spawn')
```

- Another multiprocessing-related change is formally going from using the "vanilla" Python3 `print()` command to using

  `sys.stdout.write(myline)`

  We add an explicit newline and explicitly flush that channel after we've written our results for each query (nice discussion around printing in parallel processings at `https://superfastpython.com/multiprocessing-print/`)

- We also need to retrieve our DNSDB API key in `make_query` -- if we try to do it just once, assigning that value to a "global" variable in our mainline code, our spawned process won't have the API key it needs:

```
def make_query(myfqdn):
    """ Make the DNSDB query for myfqdn """

    # global variables aren't passed to spawned processes, so we can't
just
    # get the API key once-and-done
    # get the DNSDB API key
    my_apikey = get_api_key()
```

Our biggest "surgery" involves replacing the original main serial processing loop with a new parallel version that builds a list of domains to be processed and then applies a parallel map function `(p.imap_unordered)` to that list:

```
    for line in stdin:
        make_query(line.strip())
    fqdns = []
    for line in stdin:
        fqdns.append(line.strip())

    # process the sites (across the various cores)
    with mp.Pool(processes) as p:
        p.imap_unordered(make_query, fqdns, chunksize=8)
        p.close()
        p.join()
```

You can see a full copy of the parallel version of the code in Python3-App-C. We did ten test runs with this code:

```
$ for i in {1..10}
```

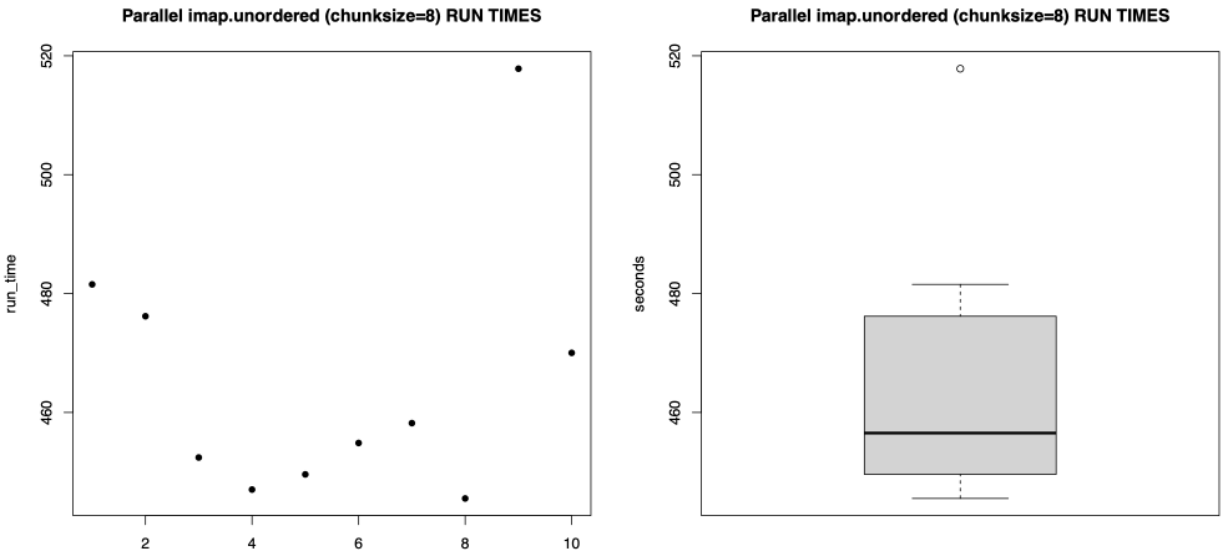
```
> do
> ./run_queries.py < all-edus.txt > all-edus-results-multiprocessing.txt
> wc -l all-edus-results-multiprocessing.txt
> done
Elapsed time:      481.536 seconds
20836814 all-edus-results-multiprocessing.txt
Elapsed time:      476.182 seconds
20837352 all-edus-results-multiprocessing.txt
Elapsed time:      452.413 seconds
20837569 all-edus-results-multiprocessing.txt
Elapsed time:      447.027 seconds
20837701 all-edus-results-multiprocessing.txt
Elapsed time:      449.576 seconds
20837810 all-edus-results-multiprocessing.txt
Elapsed time:      454.866 seconds
20837894 all-edus-results-multiprocessing.txt
Elapsed time:      458.208 seconds
20838004 all-edus-results-multiprocessing.txt
Elapsed time:      445.533 seconds
20838103 all-edus-results-multiprocessing.txt
Elapsed time:      517.817 seconds
20838390 all-edus-results-multiprocessing.txt
Elapsed time:      470.027 seconds
20838672 all-edus-results-multiprocessing.txt
```

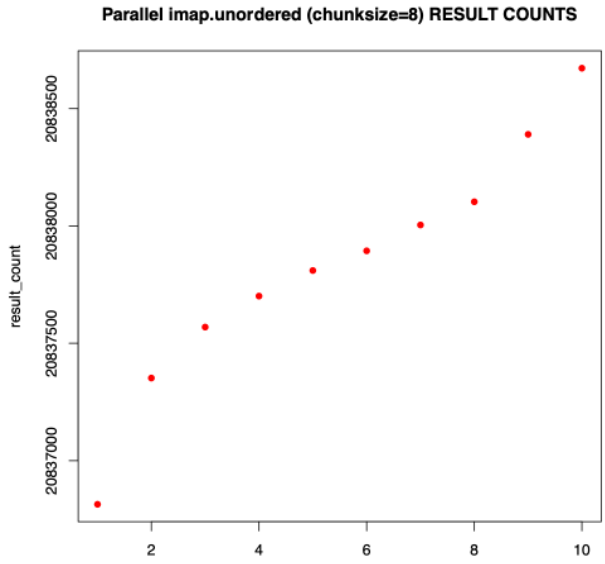
A full copy of the R code is provided in R-App-A. Looking at the results in R as we did for the serial code, we see:

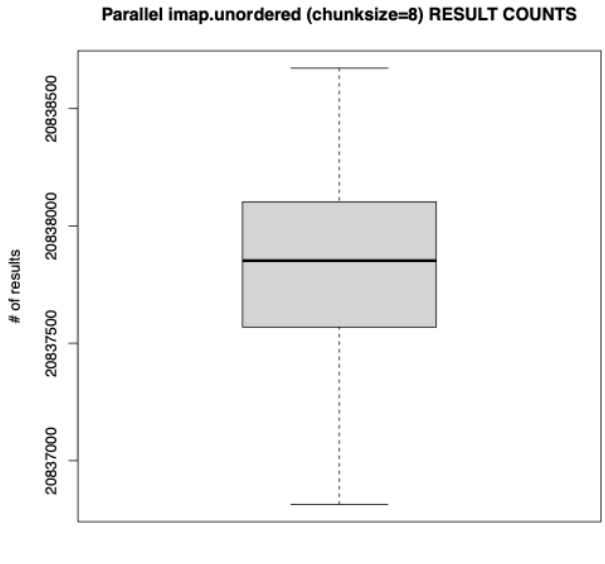
```
$ R -q < univariate-parallel-1.R
[…]
> mean(run_time)
[1] 465.3185
> sd(run_time)
[1] 22.25989
> min(run_time)
[1] 445.533
> median(run_time)
[1] 456.537
> max(run_time)
[1] 517.817
> max(run_time) - min(run_time)
[1] 72.284
[…]
> mean(result_count)
```

```
[1] 20837831
> sd(result_count)
[1] 524.6543
> min(result_count)
[1] 20836814
> median(result_count)
[1] 20837852
> max(result_count)
[1] 20838672
> max(result_count) - min(result_count)
[1] 1858
[…]
```

**Parallel run times of ~465 seconds, returning around 20.8 million results? We like that performance compared to our serial code (which returned roughly the same number of results but took an average of about 4,161 seconds). This means our parallel code delivered a speedup of 4,161.092/465.318=8.9X**






The number of results returned appear to increase monotonically, run-over-run.

## (8) Other Multiprocessing Options

The parallel code shown above uses `imap_unordered`, the most "sophisticated" ("complicated?") of **three** parallel versions of the `multiprocessing` library `"map"` function:

| | ***output returned in the same order as input*** | ***output returned "unordered" (e.g., as completed)*** |
|---|---|---|
| ***map*** | ***map:*** apply the function to **ALL** arguments from the supplied list, returning the results **in order** when ALL tasks have finished running | |
| ***imap*** | ***imap:*** lazy version of map, taking arguments from the supplied list **as required** (rather than all at once), returning the results **in the as-input order** as soon as the results are available | ***imap_unordered:*** lazy version of imap, taking arguments from the supplied list **as required** (rather than all at once), returning the results **as soon as results are available, irrespective of the input order** |

We selected the `imap_unordered "map"` option (instead of `map` or `imap`) because:

(a) We're potentially processing A LOT of data, and
(b) The results are voluminous and will take a while to transfer (so we'll want to start getting those results downloading as soon as possible).

The (optional) "`chunksize`" parameter determines how the list (the "iterable") that's passed to the various parallel map routines gets "chopped up." If `chunksize` is omitted:

- For `map`, `chunksize` defaults to the length of the iterable divided by (the size of the processor pool * 4), rounded up if non-integer. So 7432/(10*4) rounds up to 186 [see

  `https://superfastpython.com/multiprocessing-pool-map-chunksize/ ]`

- For `imap` and `imap_unordered`, if `chunksize` is omitted, chunksize defaults to 1.

We explored the impact of varying chunksize for our code, and found the code to be relatively equally performant with values ranging from `chunksize=1` to `chunksize=16`, with an apparent optimum at or around `chunksize=8`. [Chunksize result evaluation code omitted here]

For thoroughness, we'll now look at the relative performance of the other two map function options (hint: we discovered it basically doesn't matter for our use case).

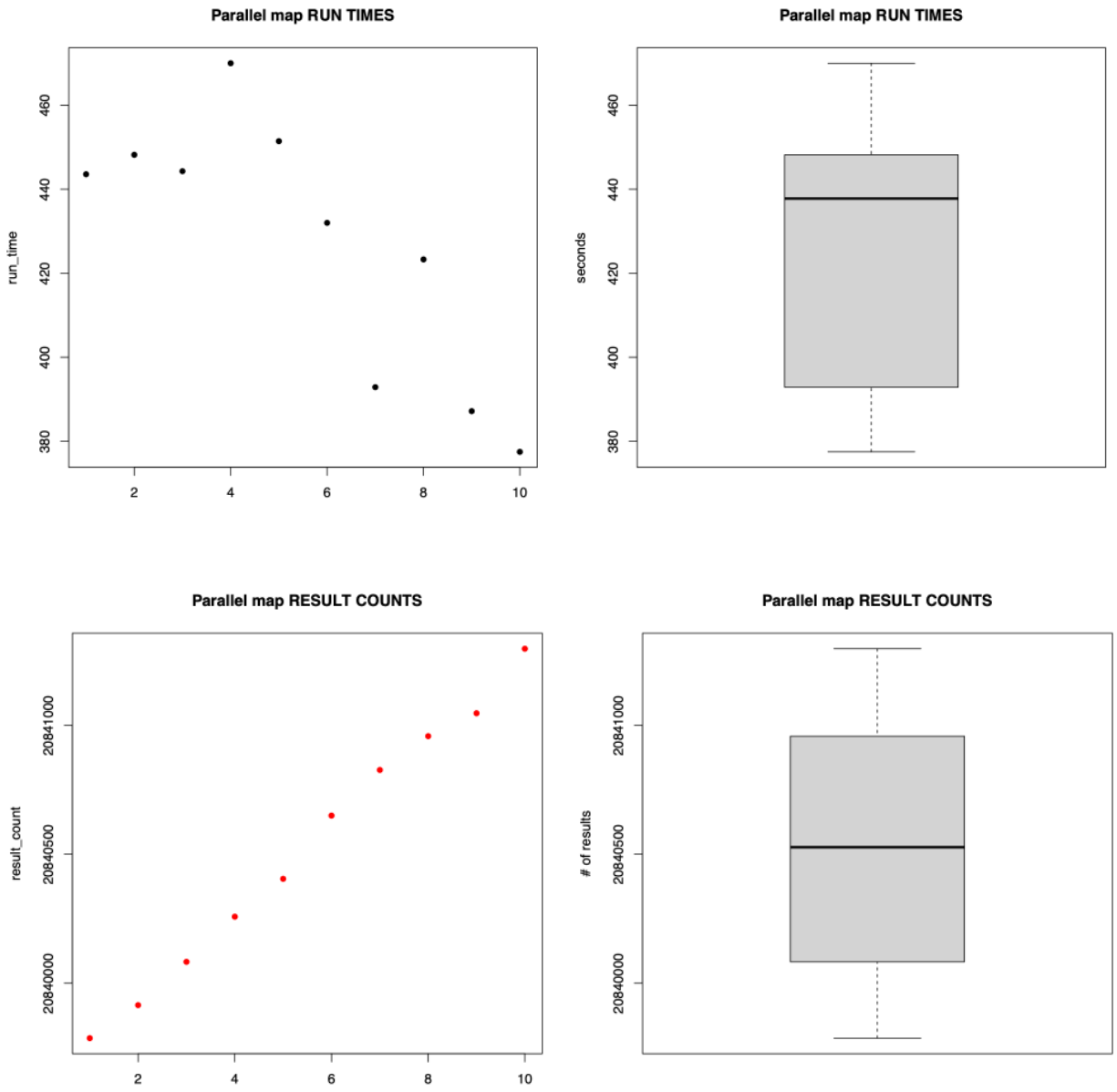
## (9) Parallel Mode: map

This run is done using `p.map`, which applies the specified function to ALL arguments from the supplied list, returning the results in order when ALL tasks have finished running. A copy of this code can be found in Python3-App-D.

```
$ for i in {1..10}
> do
> ./run_queries_map.py < all-edus.txt >
all-edus-results-multiprocessing-map.txt
> wc -l all-edus-results-multiprocessing-map.txt
> done
Elapsed time:      443.553 seconds
20839787 all-edus-results-multiprocessing-map.txt
Elapsed time:      448.175 seconds
20839915 all-edus-results-multiprocessing-map.txt
Elapsed time:      444.279 seconds
20840083 all-edus-results-multiprocessing-map.txt
Elapsed time:      469.960 seconds
20840258 all-edus-results-multiprocessing-map.txt
Elapsed time:      451.423 seconds
20840405 all-edus-results-multiprocessing-map.txt
Elapsed time:      432.002 seconds
20840650 all-edus-results-multiprocessing-map.txt
```

```
Elapsed time:      392.891 seconds
20840827 all-edus-results-multiprocessing-map.txt
Elapsed time:      423.294 seconds
20840958 all-edus-results-multiprocessing-map.txt
Elapsed time:      387.184 seconds
20841047 all-edus-results-multiprocessing-map.txt
Elapsed time:      377.515 seconds
20841297 all-edus-results-multiprocessing-map.txt

$ R -q < univariate-parallel-map.R
[…]
> length(run_time)
[1] 10
> mean(run_time)
[1] 427.0276
> sd(run_time)
[1] 31.08991
> min(run_time)
[1] 377.515
> median(run_time)
[1] 437.7775
> max(run_time)
[1] 469.96
> max(run_time) - min(run_time)
[1] 92.445
[…]
> length(result_count)
[1] 10
> mean(result_count)
[1] 20840523
> sd(result_count)
[1] 512.1339
> min(result_count)
[1] 20839787
> median(result_count)
[1] 20840528
> max(result_count)
[1] 20841297
> max(result_count) - min(result_count)
[1] 1510
[…]
```

Looking at the associated graphs, they look like:



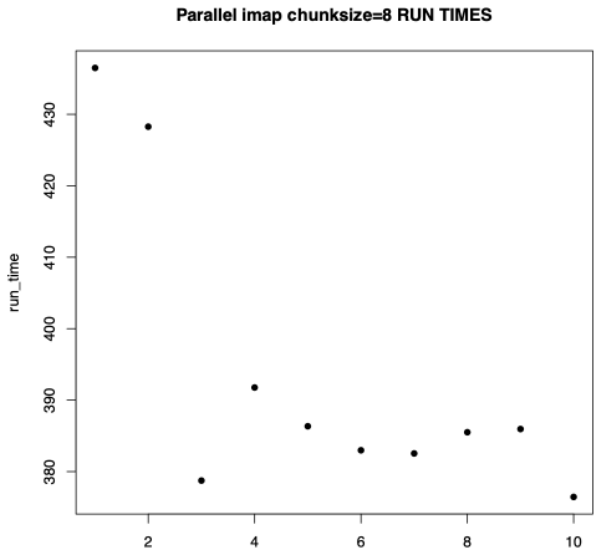

## (10) Parallel Mode: imap

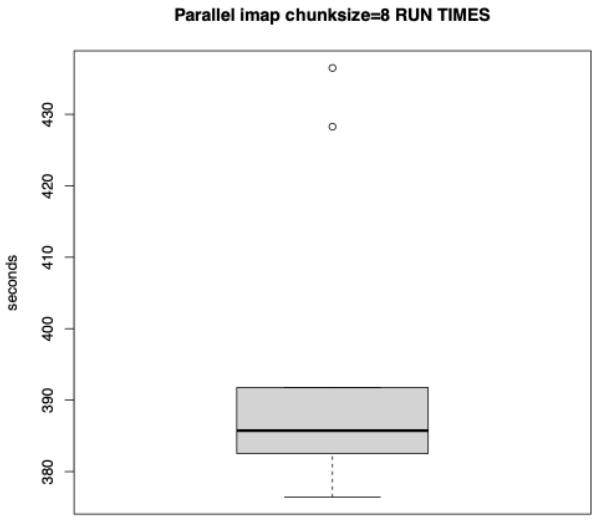
Now we retest with the "lazy" version of `p.map`, aka `"p.imap."` It takes arguments from the supplied list as required (rather than all at once), returning the results in the as-input order as soon as the results are available. This code can be found in Python3-App-E.

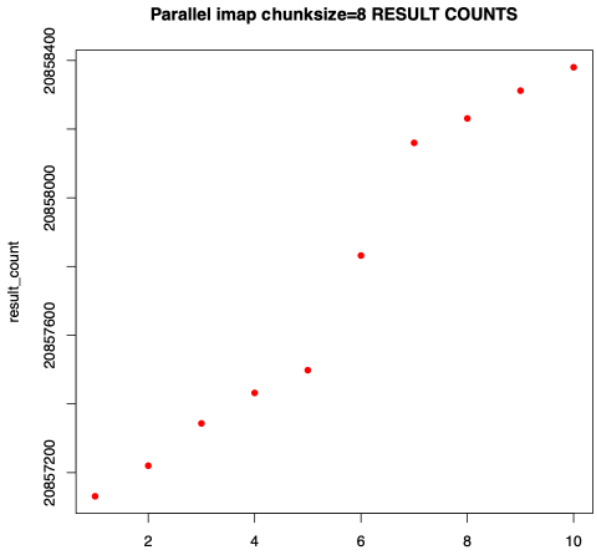
```
$ for i in {1..10}
> do
> ./run_queries_imap.py < all-edus.txt > all-edus-results-imap.txt
```

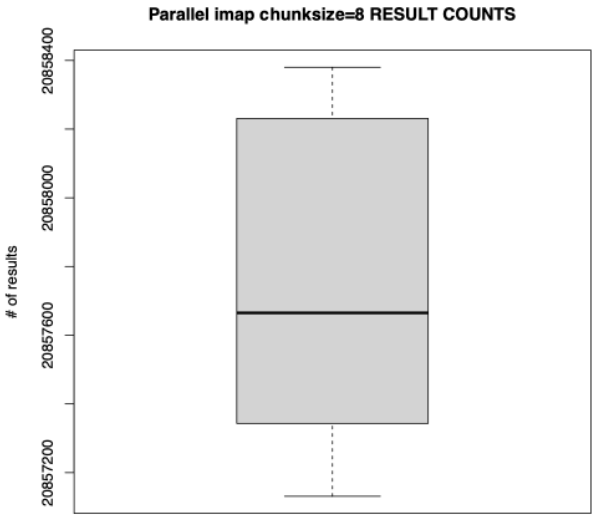
```
> wc -l all-edus-results-imap.txt
> done
Elapsed time:      436.497 seconds
20857131 all-edus-results-imap.txt
Elapsed time:      428.283 seconds
20857220 all-edus-results-imap.txt
Elapsed time:      378.741 seconds
20857343 all-edus-results-imap.txt
Elapsed time:      391.764 seconds
20857432 all-edus-results-imap.txt
Elapsed time:      386.347 seconds
20857498 all-edus-results-imap.txt
Elapsed time:      382.980 seconds
20857832 all-edus-results-imap.txt
Elapsed time:      382.547 seconds
20858160 all-edus-results-imap.txt
Elapsed time:      385.509 seconds
20858231 all-edus-results-imap.txt
Elapsed time:      385.960 seconds
20858312 all-edus-results-imap.txt
Elapsed time:      376.456 seconds
20858380 all-edus-results-imap.txt

$ R -q < univariate-parallel-imap.R
[…]
> length(run_time)
[1] 10
> mean(run_time)
[1] 393.5084
> sd(run_time)
[1] 21.00912
> min(run_time)
[1] 376.456
> median(run_time)
[1] 385.7345
> max(run_time)
[1] 436.497
> max(run_time) - min(run_time)
[1] 60.041
[…]
> length(result_count)
[1] 10
> mean(result_count)
[1] 20857754
```

```
> sd(result_count)
[1] 484.5519
> min(result_count)
[1] 20857131
> median(result_count)
[1] 20857665
> max(result_count)
[1] 20858380
> max(result_count) - min(result_count)
[1] 1249
```




## (11) Parallel Mode: threaded version

Python3 threading is potentially less powerful than Python3 multiprocessing (since all Python3 threads run on a single processor). Nonetheless, since testing threading basically requires only changing a single line from our Python3 multiprocessing `imap_unordered` code, and sometimes running with threading rather than multiprocessing may be preferred as simpler or less system impactful, we'll show that option here for the sake of completeness. To do threading, we change "`import`

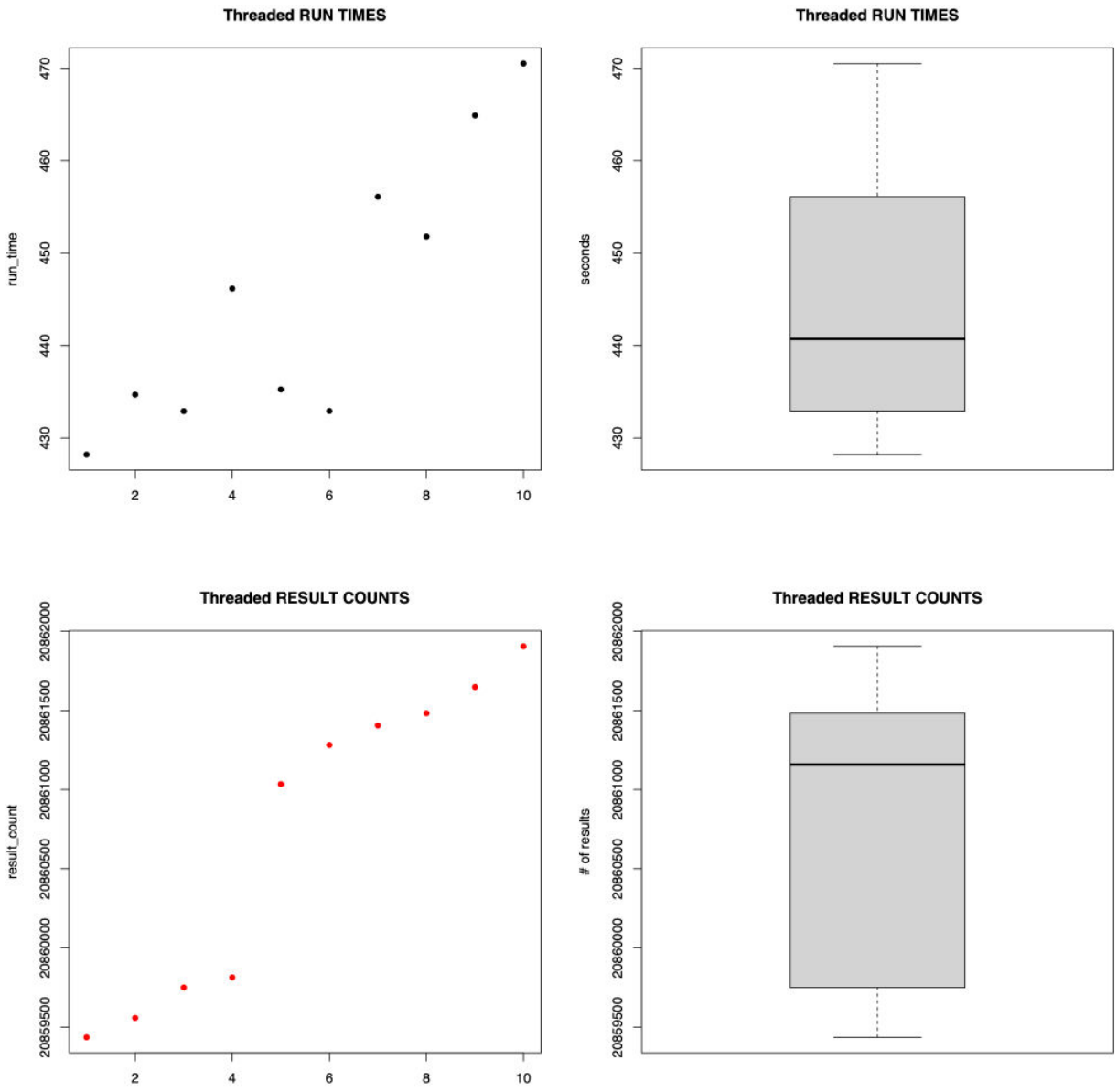
multiprocessing as mp" to
"import multiprocessing.dummy as mp" For more on that library, see:
https://docs.python.org/3/library/multiprocessing.html#module-multiprocessing.dumm

The natural question at this point is, "How does going from multiprocessing to threading impact performance?"

```
$ for i in {1..10}; do ./run_queries_threaded_mode.py < all-edus.txt >
all-edus-threaded.txt ; wc -l all-edus-threaded.txt; done
Elapsed time:      428.203 seconds
20859436 all-edus-threaded.txt
Elapsed time:      434.683 seconds
20859558 all-edus-threaded.txt
Elapsed time:      432.897 seconds
20859750 all-edus-threaded.txt
Elapsed time:      446.154 seconds
20859814 all-edus-threaded.txt
Elapsed time:      435.250 seconds
20861034 all-edus-threaded.txt
Elapsed time:      432.910 seconds
20861282 all-edus-threaded.txt
Elapsed time:      456.082 seconds
20861405 all-edus-threaded.txt
Elapsed time:      451.791 seconds
20861482 all-edus-threaded.txt
Elapsed time:      464.881 seconds
20861648 all-edus-threaded.txt
Elapsed time:      470.497 seconds
20861905 all-edus-threaded.txt

$ R -q < univariate-threaded.R
[…]
> mean(run_time)
[1] 445.3348
> sd(run_time)
[1] 14.87273
> min(run_time)
[1] 428.203
> median(run_time)
[1] 440.702
> max(run_time)
[1] 470.497
```

```
> max(run_time) - min(run_time)
[1] 42.294
[…]
> mean(result_count)
[1] 20860731
> sd(result_count)
[1] 971.1994
> min(result_count)
[1] 20859436
> median(result_count)
[1] 20861158
> max(result_count)
[1] 20861905
> max(result_count) - min(result_count)
[1] 2469
[…]
```

Now let's look at the associated graphs…




## (12) Parallel Mode: asyncio version

Another option for parallelizing code involves using asynchronous (or "non-blocking") code. There are many different asynchronous programming approaches. For this code, we're going to use semaphores. Semaphores control the number of concurrent sessions running at the same time. Since DNSDB allows up to 10 concurrent sessions, we'll initialize our semaphore count to 10:

```
# DNSDB API allows up to 10 concurrent query streams
max_sessions=10
```

```
    semaphore = asyncio.Semaphore(value=max_sessions)
```

Asynchronous tasks get queued up with:

```
        lines = []
        for line in stdin:
            lines.append(get_and_dump_dnsdb_results(line.strip(),
    semaphore))
        await asyncio.gather(*lines)
```

The following code block defines a coroutine function (see `https://docs.python.org/3/library/asyncio-task.html`) called `get_and_dump_dnsdb_results` –

```
    async def get_and_dump_dnsdb_results(myfqdn, semaphore) -> None:
        """ wrapper to combine the async dnsdb query and async results
    output """
        content = await make_query(myfqdn, semaphore)
        await write_to_stdout(content)
```

Note that both of the calls in the preceding are prefixed by "await." "await" has been called "the very core of asynchronous code." ("Python Asyncio Part 2 – Awaitables, Tasks, and Futures", `https://bbc.github.io/cloudfit-public-docs/asyncio/asyncio-part-2.html`)

Also note the use of Python3 "annotations" (e.g., "`-> None`"). If you're not familiar with Python3 annotations, see the excellent discussion at `https://stackoverflow.com/questions/14379753/what-does-mean-in-python-function-definitions` and `https://peps.python.org/pep-3107/`

But what do the `make_query` and the `write_to_stdout` functions do? They're very similar to the `make_query` function we've previously used, except that these use the `aiohttp` library (an asynchronous http library), and wait for an available semaphore before running. When the DNSDB query they're processing finishes, that semaphore then gets released for use by a new task.

```
     async def make_query(myfqdn, semaphore) -> bytes:
        """ Make the DNSDB query for myfqdn """
        # get the DNSDB API key
        my_apikey = get_api_key()

        myheaders = {'X-API-Key': my_apikey, 'Accept':
    'application/jsonl'}
        url = "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/" +
    myfqdn + \
```

```
            "?limit=1000000&time_last_after=-2592000"

        # make an aiohttp call to DNSDB (we're using this instead of
    "requests")
        async with aiohttp.ClientSession(headers=myheaders) as session:
            await semaphore.acquire()
            async with session.get(url, timeout=21600) as resp:
                content = await resp.read()
                semaphore.release()
                return content
```

Our `write_to_stdout` function is also fairly typical, except it is ALSO asynchronous:

```
    async def write_to_stdout(content: bytes) -> None:
        """ return the results asynchronously """

        # convert from bytes to string
        content2 = content.decode()
        stripped_results = remove_saf_entries(content2)
        for mylines in stripped_results:
            oneline = print_detailed_bits(mylines)
            if oneline is not None:
                sys.stdout.write(oneline)
            sys.stdout.flush()
```

The only other async-specific bits relate to processing the ready to be processed task list:

```
        lines = []
        for line in stdin:
            lines.append(get_and_dump_dnsdb_results(line.strip(),
    semaphore))
```

The full asyncio code can be seen in Python3-App-G.

First, however, let's be sure we relax the macOS bash shell's default open file descriptor limits:

```
$ ulimit -S -n 50000
```

We can then run 10 repetitions of that code:

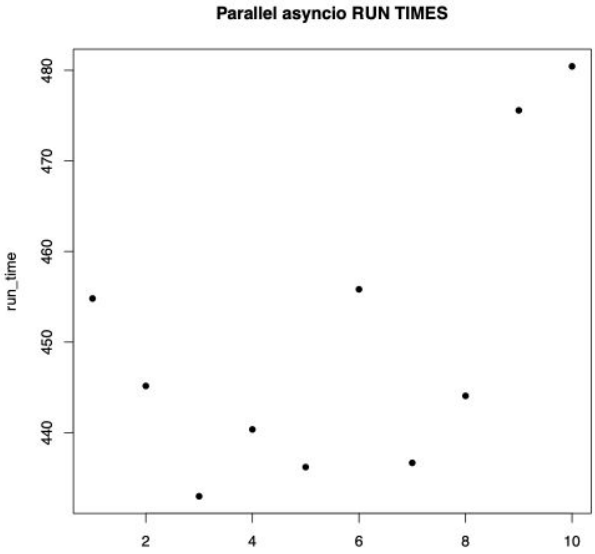
```
$ for i in {1..10}
> do
> ./run_asyncio.py < all-edus.txt > asyncio-results.txt
> wc -l asyncio-results.txt
```

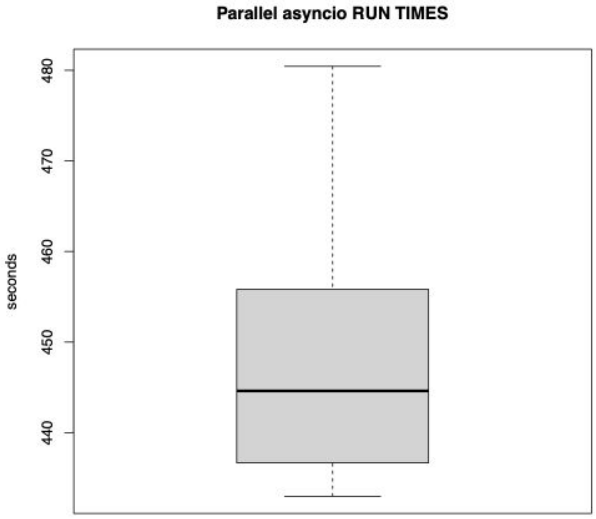
```
> done
Elapsed time:      454.811 seconds
20891690 asyncio-results.txt
Elapsed time:      445.150 seconds
20892773 asyncio-results.txt
Elapsed time:      432.975 seconds
20893705 asyncio-results.txt
Elapsed time:      440.346 seconds
20894425 asyncio-results.txt
Elapsed time:      436.196 seconds
20895066 asyncio-results.txt
Elapsed time:      455.818 seconds
20895787 asyncio-results.txt
Elapsed time:      436.662 seconds
20896658 asyncio-results.txt
Elapsed time:      444.055 seconds
20897005 asyncio-results.txt
Elapsed time:      475.578 seconds
20898307 asyncio-results.txt
Elapsed time:      480.435 seconds
20898802 asyncio-results.txt
```

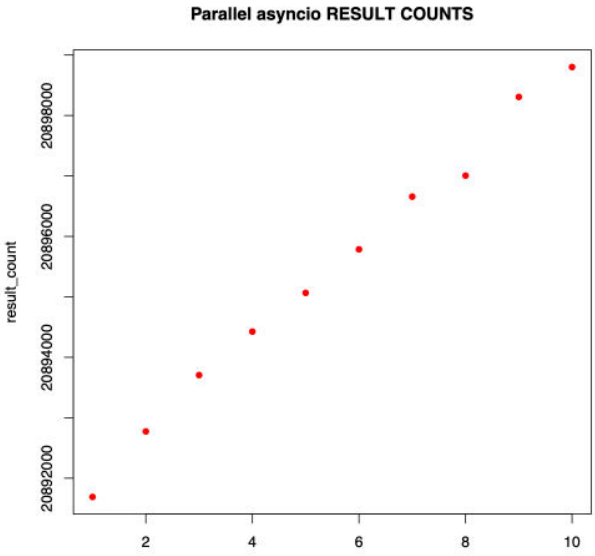
Taking those results into R, we see:

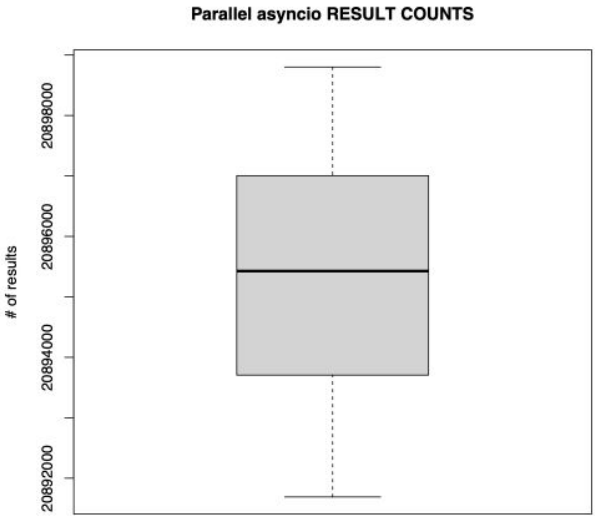
```
$ R -q < univariate-parallel-asyncio.R
[…]
> length(run_time)
[1] 10
> mean(run_time)
[1] 450.2026
> sd(run_time)
[1] 16.49066
> min(run_time)
[1] 432.975
> median(run_time)
[1] 444.6025
> max(run_time)
[1] 480.435
> max(run_time) - min(run_time)
[1] 47.46
[…]
> length(result_count)
[1] 10
> mean(result_count)
```

```
[1] 20895422
> sd(result_count)
[1] 2331.625
> min(result_count)
[1] 20891690
> median(result_count)
[1] 20895426
> max(result_count)
[1] 20898802
> max(result_count) - min(result_count)
[1] 7112
[…]
```


Parallel asyncio RUN TIMES
run_time
480
470
460
450
440
2
4
6
8
10


Parallel asyncio RUN TIMES
seconds
480
470
460
450
440


Parallel asyncio RESULT COUNTS
result_count
20898000
20896000
20894000
20892000
2
4
6
8
10


Parallel asyncio RESULT COUNTS
# of results
20898000
20896000
20894000
20892000

## (13) Parallel Mode, Cythonized Asyncio Code

For example, what if we try Cython `(https://cython.org/)` instead? Is it faster? Let's check. We begin by running Cython to convert our Python3 code to "C" source code:

```
$ cython --embed -o run_asyncio.c run_asyncio.py
```

Now we compile that C code. (The specific include and library paths on your macOS system may vary from the author's systems to yours, but this should at least give you an idea of where to look)

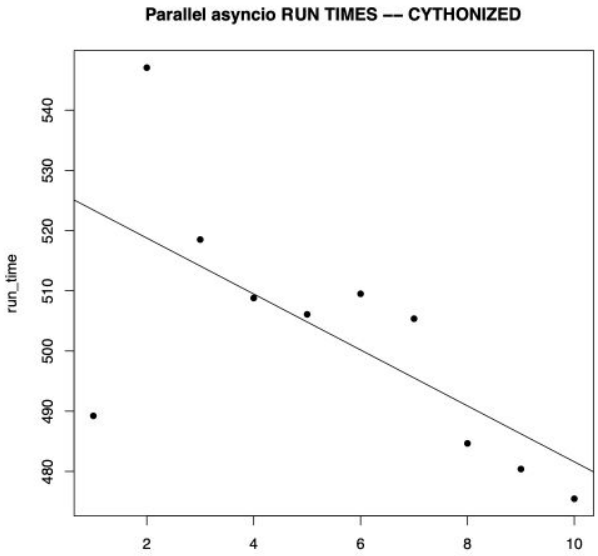
```
$ clang -I
/Library/Frameworks/Python.framework/Versions/3.10/include/python3.10 \
-L /Library/Frameworks/Python.framework/Versions/3.10/lib/
-lpython3.10 run_asyncio.c \
-o run_async_cythonized
$ ls -l run_async_cythonized
-rwxr-xr-x 1 joe staff 224817 Jan  7 19:20 run_async_cythonized
$ ./run_async_cythonized < all-edus.txt > cythonized-results.txt
Elapsed time:      473.467 seconds
$ wc -l cythonized-results.txt
20942635 cythonized-results.txt$ for i in {1..10}
```

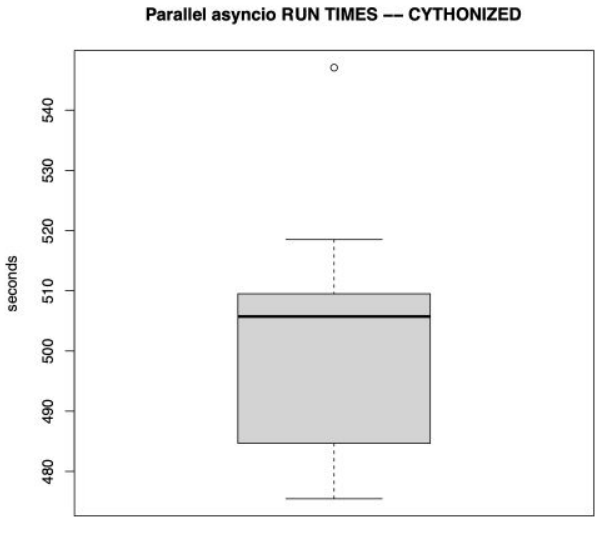
Running ten repetitions, we see:

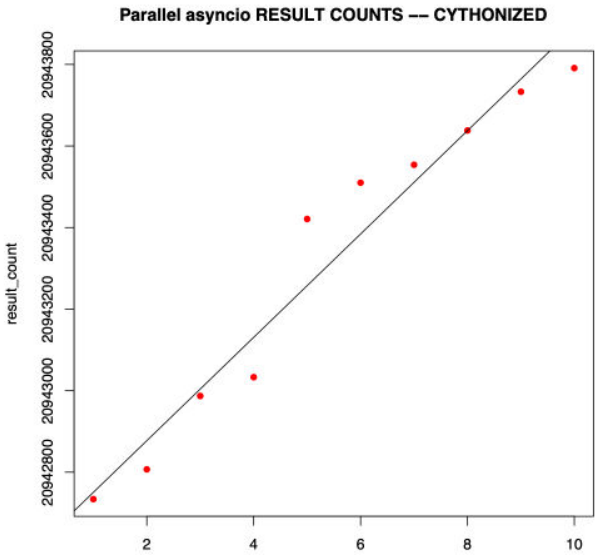
```
$ for i in {1..10}
> do
> ./run_async_cythonized < all-edus.txt > cythonized-results.txt
> wc -l cythonized-results.txt
> done
Elapsed time:      489.227 seconds
20942734 cythonized-results.txt
Elapsed time:      547.059 seconds
20942807 cythonized-results.txt
Elapsed time:      518.502 seconds
20942987 cythonized-results.txt
Elapsed time:      508.796 seconds
20943033 cythonized-results.txt
Elapsed time:      506.089 seconds
20943421 cythonized-results.txt
Elapsed time:      509.494 seconds
20943510 cythonized-results.txt
```

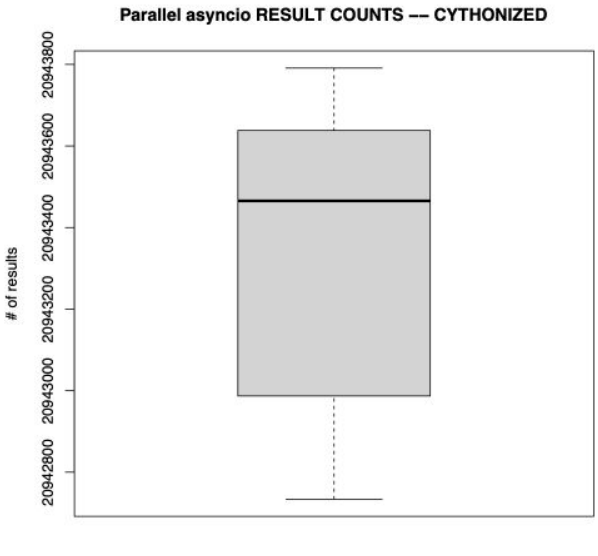
```
Elapsed time:      505.359 seconds
20943554 cythonized-results.txt
Elapsed time:      484.640 seconds
20943638 cythonized-results.txt
Elapsed time:      480.389 seconds
20943733 cythonized-results.txt
Elapsed time:      475.454 seconds
20943791 cythonized-results.txt
$ R -q < univariate-parallel-asyncio-cythonized.R
[…]
> length(run_time)
[1] 10
> mean(run_time)
[1] 502.5009
> sd(run_time)
[1] 21.25693
> min(run_time)
[1] 475.454
> median(run_time)
[1] 505.724
> max(run_time)
[1] 547.059
> max(run_time) - min(run_time)
[1] 71.605
>
> run_time_model <- lm(run_time ~ x)
> summary(run_time_model)
Call:
lm(formula = run_time ~ x)
Residuals:
    Min      1Q  Median      3Q     Max
-34.188  -6.061   0.294   8.083  28.291
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  528.063     11.544  45.742 5.76e-11 ***
x             -4.648      1.861  -2.498   0.0371 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 16.9 on 8 degrees of freedom
Multiple R-squared:  0.4382,     Adjusted R-squared:  0.368
F-statistic:  6.24 on 1 and 8 DF,  p-value: 0.03706
[…]
> length(result_count)
[1] 10
```

```
> mean(result_count)
[1] 20943321
> sd(result_count)
[1] 393.6168
> min(result_count)
[1] 20942734
> median(result_count)
[1] 20943466
> max(result_count)
[1] 20943791
> max(result_count) - min(result_count)
[1] 1057
>
> result_count_model <- lm(result_count ~ x)
> summary(result_count)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
20942734 20942998 20943466 20943321 20943617 20943791
[…]
```


Parallel asyncio RUN TIMES -- CYTHONIZED
run_time
480
490
500
510
520
530
540
2
4
6
8
10


Parallel asyncio RUN TIMES -- CYTHONIZED
seconds
480
490
500
510
520
530
540


Parallel asyncio RESULT COUNTS -- CYTHONIZED
result_count
20942800
20943000
20943200
20943400
20943600
20943800
2
4
6
8
10

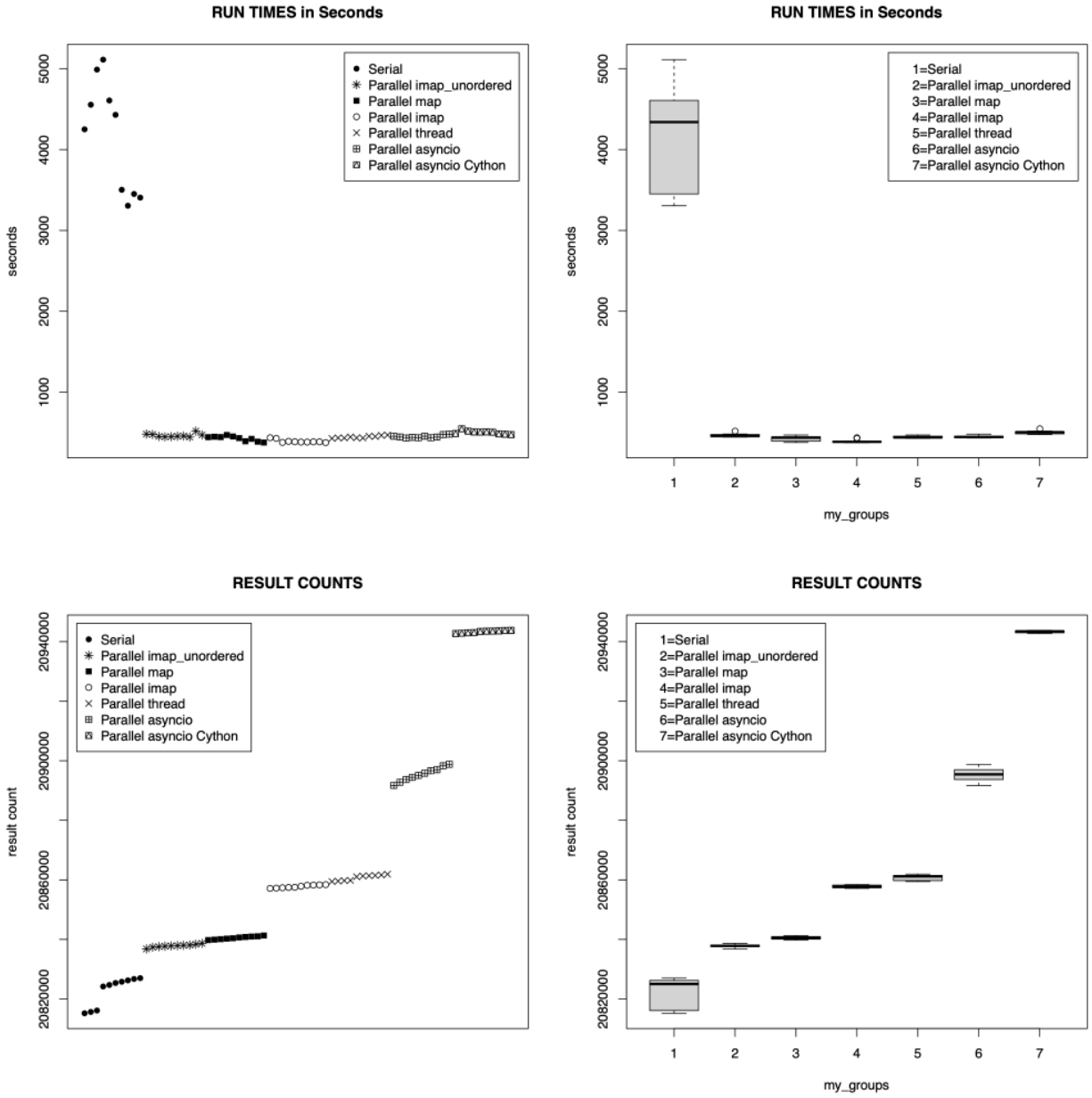

Parallel asyncio RESULT COUNTS -- CYTHONIZED
# of results
20942800
20943000
20943200
20943400
20943600
20943800

# (14) Wrapping Up Our Statistical Performance Testing

We can wrap up the seven preceding experimental conditions with the following table and graphs:

| (N=10 repetitions per treatment) | Run Time | | Result Counts | |
|---|---|---|---|---|
| | **Mean** | **SD** | **Mean** | **SD** |
| **Serial** | **4,161.092** | 688.536 | **20,822,695** | 4,940.156 |
| **Parallel imap_unordered** | 465.318 | 22.260 | 20,837,831 | 524.654 |
| **Parallel map** | 427.027 | 31.090 | 20,840,523 | 512.133 |
| **Parallel imap** | **393.508** | 21.009 | 20,857,754 | 484.552 |
| **Parallel thread** | 445.335 | 14.873 | 20,860,731 | 971.200 |
| **Parallel asyncio** | 450.203 | 16.491 | 20,895,422 | 2,331.625 |
| **Parallel asyncio Cythonized** | 502.501 | 21.257 | **20,943,321** | 393.617 |

Obviously, any of the parallel approach trump the serial approach when it comes to speed:

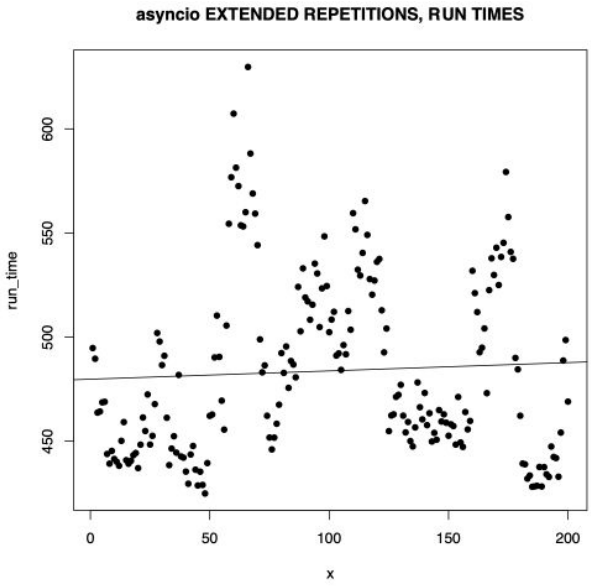


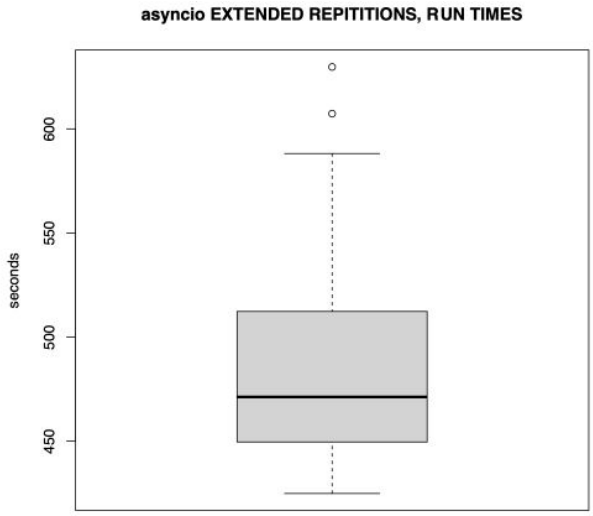

## (15) Extended Runs Confirming Increases in Results Run-over-Run

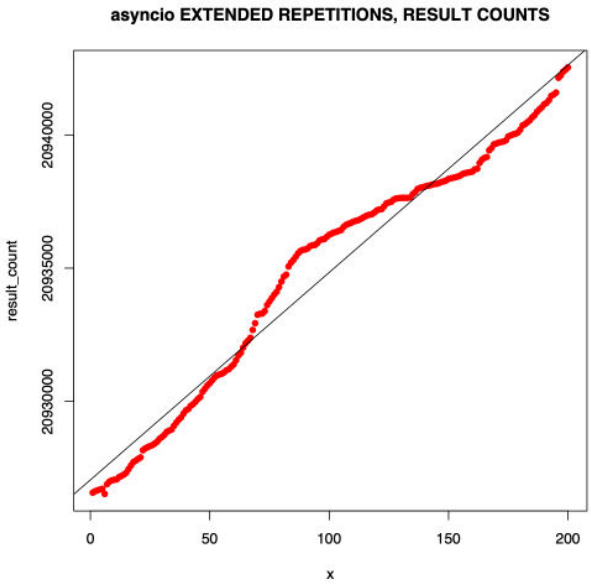
The number of results increased from run-to-run with the same set of queries. We were curious if that pattern would "plateau" after a certain number of runs. To test this, we did 200 new runs of the previously-discussed asyncio code. If the increase in results did occur, it didn't happen after 200 repetitions, as shown in the red plot below.

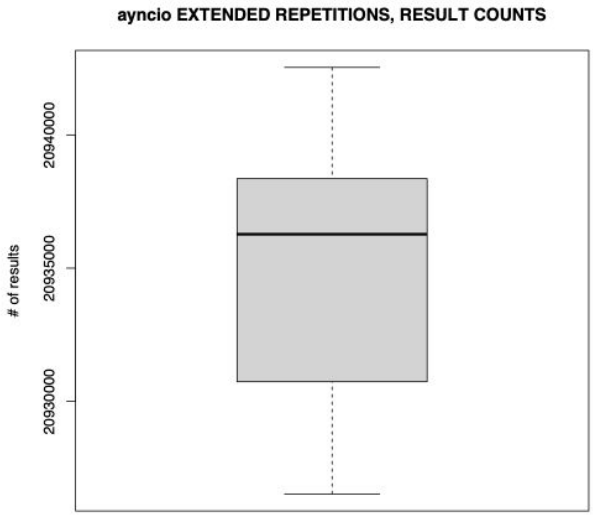
```
$ R -q < extra-univariate.R
[…]
> length(run_time)
[1] 200
> mean(run_time)
[1] 483.7782
> sd(run_time)
[1] 42.34691
> min(run_time)
[1] 424.845
> median(run_time)
[1] 471.2195
> max(run_time)
[1] 629.753
> max(run_time) - min(run_time)
[1] 204.908
> sum(run_time)
[1] 96755.65
>
> run_time_model <- lm(run_time ~ x)
> summary(run_time_model)
Call:
lm(formula = run_time ~ x)
Residuals:
   Min     1Q Median     3Q    Max
-59.12 -34.03 -12.61  28.31 147.35
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 479.75785    6.01743  79.728   <2e-16 ***
x             0.04000    0.05192   0.771    0.442
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 42.39 on 198 degrees of freedom
Multiple R-squared:  0.00299,   Adjusted R-squared:  -0.002046
F-statistic: 0.5937 on 1 and 198 DF,  p-value: 0.4419
[…]
> length(result_count)
[1] 200
> mean(result_count)
[1] 20934878
> sd(result_count)
[1] 4578.842
> min(result_count)
[1] 20926528
```

```
> median(result_count)
[1] 20936275
> max(result_count)
[1] 20942533
> max(result_count) - min(result_count)
[1] 16005
> sum(result_count)
[1] 4186975602
>
> result_count_model <- lm(result_count ~ x)
> summary(result_count_model)
Call:
lm(formula = result_count ~ x)
Residuals:
   Min     1Q Median     3Q    Max
-995.6 -710.2 -316.4  709.8 1739.8
Coefficients:
             Estimate Std. Error   t value Pr(>|t|)
(Intercept) 2.093e+07  1.170e+02 178810.26   <2e-16 ***
x           7.782e+01  1.010e+00     77.07   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 824.5 on 198 degrees of freedom
Multiple R-squared:  0.9677,    Adjusted R-squared:  0.9676
F-statistic:  5940 on 1 and 198 DF,  p-value: < 2.2e-16
```


asyncio EXTENDED REPETITIONS, RUN TIMES
run_time
x
450
500
550
600
0
50
100
150
200


asyncio EXTENDED REPITITIONS, RUN TIMES
seconds
450
500
550
600


asyncio EXTENDED REPETITIONS, RESULT COUNTS
result_count
x
20930000
20935000
20940000
0
50
100
150
200


ayncio EXTENDED REPETITIONS, RESULT COUNTS
# of results
20930000
20935000
20940000

# Part IV. Some Approaches That Didn't Help Improve DNSDB API Throughput

> *"Everything is an experiment."*
> *Issey Miyake*

**In part IV, we'll consider some alternatives that often help improve throughput, but which actually turn out to *NOT* be particularly helpful when it comes to improving DNSDB API throughput.** We mention them here to help those potentially considering these approaches, so you can at least have the benefit of our experimentation, if not the benefit of a successful additional approach.

- ***Use of P vs E Cores:*** While running the Mac's UtilitiesActivity Monitor, it is clear that the "Efficiency" ("E") cores are used at times even when the "Performance" ("P") cores are more-or-less idle.

  Forcing use of the "Performance" cores is possible, but does not appear to improve throughput.

- ***Use of the Apple Silicon M1 GPU and NPU Cores:*** In addition to 4 "Performance" cores and 4 "Efficiency" cores in the Apple Silicon M1 CPU, there are also 8 GPU cores capable of delivering an estimated 2.56 GPU TFLOPS, and a 16 core "neural engine." These would normally be a potential target for parallelization.

  Unfortunately, for our workload, offloading to the GPU or the "neural engine" doesn't appear to be a productive option.

- ***HTTP Compression?*** We're downloading a significant number of results, so, a natural thought is, "Perhaps we can **enable compression** to reduce the size of our downloads?"

  Sadly, the DNSDB API server does not currently support HTTP compression.

We'll now explicitly consider each of these points.

## (16) Compression?

Many web servers can support delivery of compressed responses. If DNSDB could compress results for users, it might greatly reduce the volume of traffic that needs to be transferred -- and the time a connection slot is "tied up" transferring results.

For example, assume we want to retrieve the website at **https://www.ucla.edu/** If we retrieve that site WITHOUT requesting use of gzip compression, we see:

```
$ curl "https://www.ucla.edu" --output "temp.html"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
100 97487    0 97487    0     0   467k      0 --:--:-- --:--:-- --:--:--
498k
```

If we retrieve the www.ucla.edu site after we've requested gzip compression, we get a FAR smaller file:

```
$ curl -H "Accept-encoding: gzip" "https://www.ucla.edu" --output
"gzipped-file.gz"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
100 17074  100 17074    0     0   166k      0 --:--:-- --:--:-- --:--:--
183k
```

We can confirm that the file we received is in fact gzipped by checking that file with the command:

```
$ gzip -l gzipped-file.gz
         compressed        uncompressed  ratio uncompressed_name
              17074               97487  82.5% gzipped-file
```

Now let's try basically the same thing with a **sample DNSDB query…**

```
$ curl -H "X-API-Key: insertYourRealAPIkeyHere" -H "Accept:
application/jsonl"
"https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/www.ucla.edu" --output
"temp.jsonl"
```

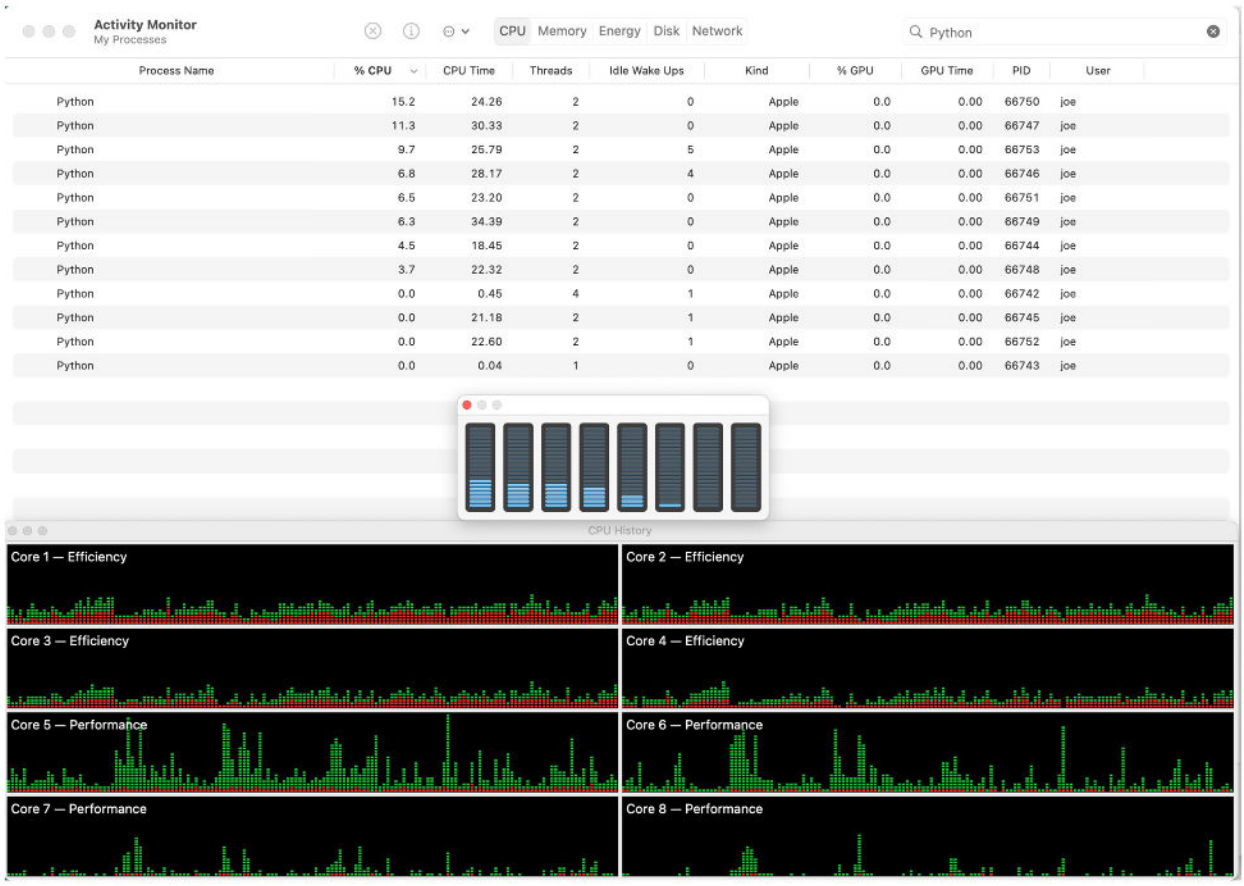
```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
100  4474    0  4474    0     0  18826      0 --:--:-- --:--:-- --:--:--
19537
```

```
$ curl -H "X-API-Key: insertYourRealAPIkeyHere" -H "Accept:
application/jsonl" -H "Accept-encoding: gzip"
"https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/www.ucla.edu" --output
"temp.jsonl"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
100  4474    0  4474    0     0  18476      0 --:--:-- --:--:-- --:--:--
19452
```

Unfortunately, there was no difference in the transferred results volume from https://api.dnsdb.info/ --compression **did NOT** occur.

## (17) Use of Performance vs Efficiency Cores

The MacBook Pro we used for these tests has a CPU that includes a mixture of "Performance" ("P") cores and "Efficiency" ("E") cores. The macOS Activity Monitor `(https://support.apple.com/guide/activity-monitor/welcome/mac)`, located under the macOS Apps "Utilities" group, makes it possible to see the extent to which the various cores are being used. For example, while running a long series of parallel `imap_unordered` multiprocessing runs, and having the Activity Monitor focus on Python processes, we noticed an apparent preference for "E" cores over "P" cores:




(Red represents system processes while green represents user processes in the above graphs). Notes on this:

- `https://apple.stackexchange.com/questions/443713/python-script-using-efficiency-cores-rather-than-performance-cores-in-m1` discusses how one can get force P cores to be used in preference to E cores, but in our experience, this does not appear to decrease runtime for our multiprocessing DNSDB queries.

- The only way we were able to "fully utilize" the macOS P cores (e.g., get "completely green" Performance Core graphs) was by using the P core preference "trick" just mentioned while also using **unbuffered** binary "write-through" I/O. (This did not appear to improve throughput, just increase CPU utilization!) If you'd like to experiment with this yourself:

```
import io, os, sys
sys.stdout = io.TextIOWrapper(open(sys.stdout.fileno(), 'wb', 0), write_through=True)
```

Source for this full unbuffering approach -- comment by Federico A. Ramponi (Oct 8, 2008 ) (edited Apr 17, 2020 by Russell Davis) which in turn credits '*"Sebastian", somewhere on the Python mailing list*'
`https://stackoverflow.com/questions/107705/disable-output-buffering`

## (18) Exploiting the Macbook Pro M1 GPU and/or Apple Neural Engine?

In addition to the 4 "E" and 4 "P" CPU cores mentioned in the preceding section, the M1 Macbook Pro also has an 8-core Graphics Processing Unit ("GPU") supporting Apple's "Metal" (see `https://developer.apple.com/metal/)`, and a 16-core NPU (or "Neural Processing Unit") called the Apple Neural Engine ("ANE"). These resources are both hard to exploit for non-machine learning-related tasks from Python3.

**The M1 GPU?** A detailed analysis of the M1 GPU can be found at:
`https://www.notebookcheck.net/Apple-M1-GPU-Benchmarks-and-Specs.503610.0.html`

- The Apple M1 GPU potentially delivers an estimated 2.56 TFLOPS.

  For comparison…

- Some top-end GPU cards for Windows or Linux desktops `(https://www.techpowerup.com/gpu-specs/):`

| | Peak TFLOP (FP32) | Price |
|---|---|---|
| NVIDIA GeForce RTX 4090 | 83 | ~$2,300 (limited availability) |
| AMD Radeon™ RX 7900 XTX | 61.42 | ~$1,483 (limited availability) |

- Current leadership class scientific parallel systems (including the DOE's 1.102 exaflop "Frontier" system), see `https://www.top500.org/lists/top500/2022/11/`

Programmatic access to the M1 GPU from Python3 is improving. See for example:

- "Core ML Framework" from Apple, `https://github.com/apple/coremltools`
- "How to Access the Metal API from Python,"

  `https://noppoman.github.io/python-metal-benchmark/Python-meets-Metal-API-on-OSX.html`

- **taichi,** `https://docs.taichi-lang.org/` Supports Metal GPU from Python. Easy to use, but requires strict typing of taichi functions & taichi kernel routine arguments and return values. Supported types are ints, floats, matrices, etc. but not non-numeric types such as strings & lists of strings `(https://docs.taichi-lang.org/api/taichi/types/)`
- **pytorch**: `https://developer.apple.com/metal/pytorch/`
  Trying the "verify" test routine shown on that page returns a user warning:

  ```
  $ ./torch-test.py
  /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/torch/_tensor_str.py:115: UserWarning: The operator 'aten::nonzero' is not currently supported on the MPS backend and will fall back to run on the CPU. This may have performance implications. (Triggered internally at /Users/runner/work/pytorch/pytorch/pytorch/aten/src/ATen/mps/MPSFallback.mm:11.)
    nonzero_finite_vals = torch.masked_select(tensor([1.], device='mps:0'))
  ```

- **tensorflow-metal plugin**: See `https://developer.apple.com/metal/tensorflow-plugin/` BUT some users report poor GPU performance (see `https://developer.apple.com/forums/thread/693678`)

**Apple's M1 ANE?** The M1 ANE is quoted as delivering 15.8 TFLOPS (see `https://machinelearning.apple.com/research/neural-engine-transformers`), but it doesn't appear to be a generally available resource. What is known/available about it includes:

- "The Neural Engine — what do we know about it?" `https://github.com/hollance/neural-engine`
- "ANETools," `https://github.com/antgroup-arclab/ANETools`
- "Apple Neural Engine Internal,"

  `https://i.blackhat.com/asia-21/Friday-Handouts/as21-Wu-Apple-Neural_Engine.pdf`

In general, the ANE is even less accessible/exploitable than the M1 GPU.

For now, we're calling the Mac Book GPU and ANE "inapplicable" for parallelization efforts related to DNSDB.

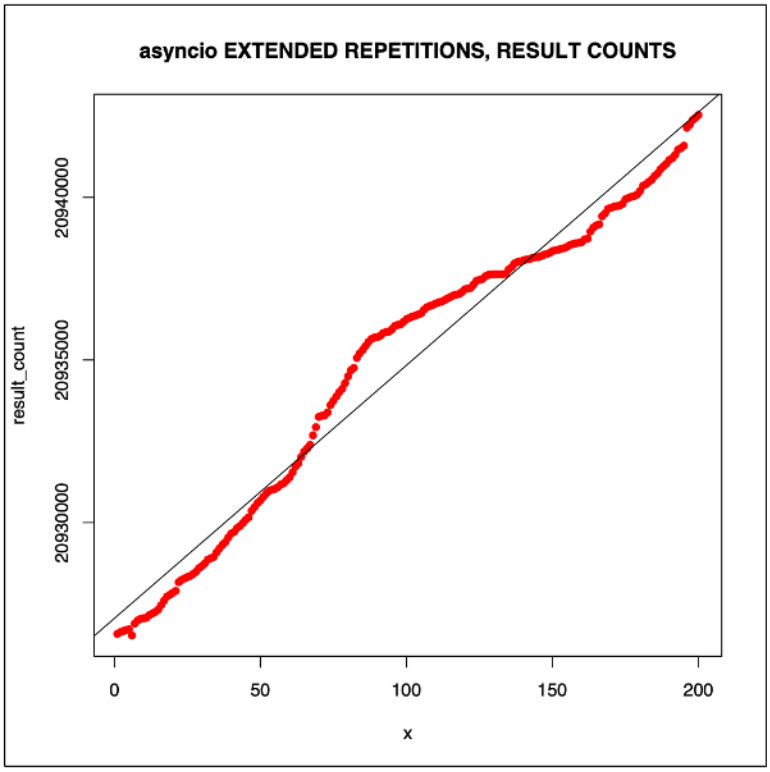
# V. Conclusion

We've covered a lot of material, but by way of conclusion, primary takeaways include:

- Parallelizing DNSDB queries makes it possible to significantly decrease total run time for jobs involving large numbers of queries.
- `dnsdbq` users can productively parallelize their queries (without need to write any code) via `dnsdbq`'s parallel batch mode.
- The effort required to take sequential DNSDB API code and convert it from serial to parallel execution is small, and the rewards can be quite large.
- Python3 coders can parallelize DNSDB API queries using Pool-based multiprocessing, threading, or asynchronous I/O approaches, all delivering roughly the same speedup of 8-9X+ given an allowed maximum of 10 parallel connections to DNSDB API.
- Review of the JSON library performance resulted in selection of `cysimdjson` as a replacement for the standard built-in `json` library.
- Python3 and "R" code run for this report has been shared for use or experimentation by the reader.

**Other insights:**

- The number of results returned from DNSDB API for batches of very large queries (particularly with time fencing) may vary from run-to-run, even when repeatedly executing the same piece of code. This variation is believed to be due to server-side OS caching and the server's fixed duration timeout for returning results (e.g., 15 minutes). An example of a longer test (involving over 200 repetitions) shows no sign of asymptotically plateauing.




- While HTTP compression could be used to reduce the number of octets that need to be transferred, HTTP compression is not currently supported by the DNSDB API server.
- Python3 is interpreted. Compiled code normally runs faster than interpreted code. However, in this case, converting Python3 code

to "C," and then compiling that C-code into an executable with `Cython`, fails to improve throughput.

- The Macbook Pro Apple Silicon M1 CPU has both four "Performance" ("P") cores and four "Efficiency" ("E") cores. Routine Python3 loads were seen to routinely use the "E" cores even when "P" cores were available. Efforts to shift load to the "P" cores did not result in a noticeable improvement in throughput
- The Macbook Pro M1 has an 8 core GPU and a 16 core NPU. Normally GPUs and NPUs would be a prime avenue for traditional matrix-focused parallelization work, but those resources are not readily exploitable from Python3 for non-machine-learning-related purposes.

# Acknowledgements

Thanks to all the colleagues who provided feedback on a draft of this report, including Mr. Aaron Gee-Clough and Dr. Sean McNee.

# Notes

Research for this paper was completed in January 2023.

# PYTHON3 CODE APPENDICES

> *"The trouble with programmers is that you can never tell what a programmer is doing until it's too late."*
> *Seymour Cray*

## Python3-App-A. Single Query

```
$ cat single_query.py
#!/usr/local/bin/python3
""" run a single DNSDB API query """
# https://requests.readthedocs.io/en/latest/
import requests
key = "insertYourRealAPIkeyHere"
myfqdn = "www.whitman.edu/A"
url = "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/" + myfqdn
myheaders = {'X-API-Key': key, 'Accept': 'application/jsonl'}
r = requests.get(url, headers=myheaders, timeout=60)
print(r.content.decode())
```

## Python3-App-B: Sequential mode

```
$ cat run_queries_serial.py
#!/usr/local/bin/python3
""" run_queries_serial.py """
import os
from pathlib import Path
from signal import signal, SIGINT
import sys
# pylint: disable=W0622
```

```
from sys import exit, stdin
import time
from time import strftime, gmtime
# https://requests.readthedocs.io/en/latest/
import requests
# https://github.com/TeskaLabs/cysimdjson
import cysimdjson
def get_api_key():
    """ Retrieve the DNSDB API key """
    api_key_file_path = os.path.join(str(Path.home()),
".dnsdb-apikey.txt")
    try:
        with open(api_key_file_path, encoding='UTF-8') as my_api_file:
            val = my_api_file.readline().strip()
    except FileNotFoundError:
        exit("DNSDB API key should be in ~/.dnsdb-apikey.txt")
    return val
def handler(_ , _2):
    """Stub routine to handle the user hitting ctrl-C"""
    exit("\nUser hit ctrl-C -- exiting")
def make_query(myfqdn):
    """ Make the DNSDB query for myfqdn """
    url = "https://api-pao.dnsdb.info/dnsdb/v2/lookup/rrset/name/" +
myfqdn + \

"?limit=1000000&time_last_after=1668377911&time_first_before=1670969911"
    myheaders = {'X-API-Key': my_apikey, 'Accept': 'application/jsonl'}
    r = requests.get(url, headers=myheaders, timeout=3600)
    # Status Code 200 == Success
    if r.status_code == 200:
        stripped_results = remove_saf_entries(r.text)
        for myfqdns in stripped_results:
            myline = print_detailed_bits(myfqdns)
            if myline is not None:
                print(myline)
    else:
        sys.stderr.write(myfqdn+" returned status
code="+r.status_code+"\n")
def print_detailed_bits(myrecord):
    """format results for display"""
    parser = cysimdjson.JSONParser()
    myrecord_json_format = parser.loads(myrecord)
    my_rrtype = myrecord_json_format['obj']['rrtype']
    # assume we're only interested in "A" records and CNAMEs
```

```
    if my_rrtype.rstrip() not in ("A", "CNAME"):
        return None
    my_rrname = myrecord_json_format['obj']['rrname']
    my_count = myrecord_json_format['obj']['count']
    my_rdata = str((myrecord_json_format['obj']['rdata']).export())
    myformat = '%y-%m-%d %H:%M'
    # time_last
    try:
        extract_tl = myrecord_json_format['obj']['time_last']
    except KeyError:
        extract_tl = myrecord_json_format['obj']['zone_time_last']
    enddatetime = strftime(myformat, gmtime(extract_tl))
    # time_first
    try:
        extract_tf = myrecord_json_format['obj']['time_first']
    except KeyError:
        extract_tf = myrecord_json_format['obj']['zone_time_first']
    startdatetime = strftime(myformat, gmtime(extract_tf))
    # format the output for display
    my_rrname = f'{my_rrname:<55}'
    my_rrtype = f'{my_rrtype:<6}'
    my_count = f'{my_count:>12,}'
    # enquoting the date times requires that we use escaped quotes
    results = f'{my_rrname} {my_rrtype} \"{enddatetime}\"
\"{startdatetime}\" {my_count} {my_rdata}'
    return results
def remove_saf_entries(mylist):
    """ Remove the Streaming API Framing Records From the Results """
    mylist2 = mylist.splitlines()
    # Strip the streaming format bookend records
    if mylist2[0] == '{"cond":"begin"}':
        mylist2.pop(0)
    if ((mylist2[-1] == '{"cond":"succeeded"}') or \
        (mylist2[-1] == '{"cond":"limited","msg":"Result limit
reached"}')):
        mylist2.pop()
    return mylist2
if __name__ == "__main__":
    # Did you forget to pipe in the sites to check?
    if stdin.isatty():
        exit("Pipe in sites to check via stdin")
    # get timing start time
    start_time = time.time()
    # handle ctrl-C interrupt cleanly
```

```
    signal(SIGINT, handler)
    # get the DNSDB API key
    my_apikey = get_api_key()
    for line in stdin:
        make_query(line.strip())
    # report elapsed time
    elapsed_time = time.time() - start_time
    elapsed_time = f'{elapsed_time:>12.3f} seconds\n'
    sys.stderr.write("Elapsed time: " + elapsed_time)
```

## Python3-App-C. Parallel mode: imap.unordered

```
$ cat run_queries.py
#!/usr/local/bin/python3
""" run_queries.py """
# pylint: disable=invalid-name, import-error, line-too-long
import multiprocessing as mp
import os
from pathlib import Path
from signal import signal, SIGINT
import sys
# pylint: disable=W0622
from sys import exit, stdin
import time
from time import strftime, gmtime
# https://github.com/TeskaLabs/cysimdjson
import cysimdjson
# https://requests.readthedocs.io/en/latest/
import requests
# DNSDB API allows up to a maximum of 10 concurrent query streams
processes = 10
def get_api_key():
    """ Retrieve the DNSDB API key """
    api_key_file_path = os.path.join(str(Path.home()),
".dnsdb-apikey.txt")
    try:
        with open(api_key_file_path, encoding='UTF-8') as my_api_file:
            val = my_api_file.readline().strip()
    except FileNotFoundError:
        exit("DNSDB API key should be in ~/.dnsdb-apikey.txt")
    return val
def handler(_ , _2):
```

```
    """Stub routine to handle the user hitting ctrl-C"""
    exit("\nUser hit ctrl-C -- exiting")
def make_query(myfqdn):
    """ Make the DNSDB query for myfqdn """
    # global variables aren't passed to spawned processes, so we can't
just
    # get the API key once-and-done
    # get the DNSDB API key
    my_apikey = get_api_key()
    url = "https://api-pao.dnsdb.info/dnsdb/v2/lookup/rrset/name/" +
myfqdn + \

"?limit=1000000&time_last_after=1668377911&time_first_before=1670969911"
    myheaders = {'X-API-Key': my_apikey, 'Accept': 'application/jsonl'}
    r = requests.get(url, headers=myheaders, timeout=3600)
    # Status Code 200 == Success
    if r.status_code == 200:
        stripped_results = remove_saf_entries(r.text)
        for myfqdns in stripped_results:
            myline = print_detailed_bits(myfqdns)
            if myline is not None:
                sys.stdout.write(myline)
        sys.stdout.flush()
    else:
        sys.stderr.write(myfqdn+" returned status
code="+r.status_code+"\n")
def print_detailed_bits(myrecord):
    """format results for display"""
    parser = cysimdjson.JSONParser()
    myrecord_json_format = parser.loads(myrecord)
    my_rrtype = myrecord_json_format['obj']['rrtype']
    # assume we're only interested in "A" records and CNAMEs
    if my_rrtype.rstrip() not in ("A", "CNAME"):
        return None
    my_rrname = myrecord_json_format['obj']['rrname']
    my_count = myrecord_json_format['obj']['count']
    my_rdata = str((myrecord_json_format['obj']['rdata']).export())
    myformat = '%y-%m-%d %H:%M'
    # time_last
    try:
        extract_tl = myrecord_json_format['obj']['time_last']
    except KeyError:
        extract_tl = myrecord_json_format['obj']['zone_time_last']
    enddatetime = strftime(myformat, gmtime(extract_tl))
```

```
    # time_first
    try:
        extract_tf = myrecord_json_format['obj']['time_first']
    except KeyError:
        extract_tf = myrecord_json_format['obj']['zone_time_first']
    startdatetime = strftime(myformat, gmtime(extract_tf))
    # format the output for display
    my_rrname = f'{my_rrname:<55}'
    my_rrtype = f'{my_rrtype:<6}'
    my_count = f'{my_count:>12,}'
    # enquoting the date times requires that we use escaped quotes
    results = f'{my_rrname} {my_rrtype} \"{enddatetime}\"
\"{startdatetime}\" {my_count} {my_rdata}\n'
    return results
def remove_saf_entries(mylist):
    """ Remove the Streaming API Framing Records From the Results """
    mylist2 = mylist.splitlines()
    # Strip the streaming format bookend records
    if mylist2[0] == '{"cond":"begin"}':
        mylist2.pop(0)
    if ((mylist2[-1] == '{"cond":"succeeded"}') or \
        (mylist2[-1] == '{"cond":"limited","msg":"Result limit
reached"}')):
        mylist2.pop()
    return mylist2
if __name__ == "__main__":
    # Did you forget to pipe in the sites to check?
    if stdin.isatty():
        exit("Pipe in sites to check via stdin")
    # get timing start time
    start_time = time.time()
    # spawn is the default for MacOS, but fork is the default for many
other Un*x systems
    # if spawn's not used, fork has implications for global variable
inheritance, etc.
    mp.set_start_method('spawn')
    # handle ctrl-C interrupt cleanly
    signal(SIGINT, handler)
    fqdns = []
    for line in stdin:
        fqdns.append(line.strip())
    # process the sites (across the various processes)
    with mp.Pool(processes) as p:
        p.imap_unordered(make_query, fqdns, chunksize=8)
```

```
        p.close()
        p.join()
    # report elapsed time
    elapsed_time = time.time() - start_time
    elapsed_time = f'{elapsed_time:>12.3f} seconds\n'
    sys.stderr.write("Elapsed time: " + elapsed_time)
```

## Python3-App-D. Parallel mode: map

```
$ cat run_queries_map.py
#!/usr/local/bin/python3
""" run_queries_map.py """
# pylint: disable=invalid-name, import-error, line-too-long
import multiprocessing as mp
import os
from pathlib import Path
from signal import signal, SIGINT
import sys
# pylint: disable=W0622
from sys import exit, stdin
import time
from time import strftime, gmtime
# https://github.com/TeskaLabs/cysimdjson
import cysimdjson
# https://requests.readthedocs.io/en/latest/
import requests
# DNSDB API allows up to a maximum of 10 concurrent query streams
processes = 10
def get_api_key():
    """ Retrieve the DNSDB API key """
    api_key_file_path = os.path.join(str(Path.home()),
".dnsdb-apikey.txt")
    try:
        with open(api_key_file_path, encoding='UTF-8') as my_api_file:
            val = my_api_file.readline().strip()
    except FileNotFoundError:
        exit("DNSDB API key should be in ~/.dnsdb-apikey.txt")
    return val
def handler(_ , _2):
    """Stub routine to handle the user hitting ctrl-C"""
    exit("\nUser hit ctrl-C -- exiting")
def make_query(myfqdn):
```

```
    """ Make the DNSDB query for myfqdn """
    # global variables aren't passed to spawned processes, so we can't
just
    # get the API key once-and-done
    # get the DNSDB API key
    my_apikey = get_api_key()
    url = "https://api-pao.dnsdb.info/dnsdb/v2/lookup/rrset/name/"+myfqdn+
\

"?limit=1000000&time_last_after=1668377911&time_first_before=1670969911"
    myheaders = {'X-API-Key': my_apikey, 'Accept': 'application/jsonl'}
    r = requests.get(url, headers=myheaders, timeout=3600)
    # Status Code 200 == Success
    if r.status_code == 200:
        stripped_results = remove_saf_entries(r.text)
        for myfqdns in stripped_results:
            myline = print_detailed_bits(myfqdns)
            if myline is not None:
                sys.stdout.write(myline)
        sys.stdout.flush()
    else:
        sys.stderr.write(myfqdn+" returned status
code="+r.status_code+"\n")
def print_detailed_bits(myrecord):
    """format results for display"""
    parser = cysimdjson.JSONParser()
    myrecord_json_format = parser.loads(myrecord)
    my_rrtype = myrecord_json_format['obj']['rrtype']
    # assume we're only interested in "A" records and CNAMEs
    if my_rrtype.rstrip() not in ("A", "CNAME"):
        return None
    my_rrname = myrecord_json_format['obj']['rrname']
    my_count = myrecord_json_format['obj']['count']
    my_rdata = str((myrecord_json_format['obj']['rdata']).export())
    myformat = '%y-%m-%d %H:%M'
    # time_last
    try:
        extract_tl = myrecord_json_format['obj']['time_last']
    except KeyError:
        extract_tl = myrecord_json_format['obj']['zone_time_last']
    enddatetime = strftime(myformat, gmtime(extract_tl))
    # time_first
    try:
        extract_tf = myrecord_json_format['obj']['time_first']
```

```
    except KeyError:
        extract_tf = myrecord_json_format['obj']['zone_time_first']
    startdatetime = strftime(myformat, gmtime(extract_tf))
    # format the output for display
    my_rrname = f'{my_rrname:<55}'
    my_rrtype = f'{my_rrtype:<6}'
    my_count = f'{my_count:>12,}'
    # enquoting the date times requires that we use escaped quotes
    results = f'{my_rrname} {my_rrtype} \"{enddatetime}\"
\"{startdatetime}\" {my_count} {my_rdata}\n'
    return results
def remove_saf_entries(mylist):
    """ Remove the Streaming API Framing Records From the Results """
    mylist2 = mylist.splitlines()
    # Strip the streaming format bookend records
    if mylist2[0] == '{"cond":"begin"}':
        mylist2.pop(0)
    if ((mylist2[-1] == '{"cond":"succeeded"}') or \
        (mylist2[-1] == '{"cond":"limited","msg":"Result limit
reached"}')):
        mylist2.pop()
    return mylist2
if __name__ == "__main__":
    # Did you forget to pipe in the sites to check?
    if stdin.isatty():
        exit("Pipe in sites to check via stdin")
    # get timing start time
    start_time = time.time()
    # spawn is the default for MacOS, but fork is the default for many
other Un*x systems
    # if spawn's not used, fork has implications for global variable
inheritance, etc.
    mp.set_start_method('spawn')
    # handle ctrl-C interrupt cleanly
    signal(SIGINT, handler)
    fqdns = []
    for line in stdin:
        fqdns.append(line.strip())
    # process the sites (across the various processes)
    with mp.Pool(processes) as p:
        p.map(make_query, fqdns, chunksize=8)
        p.close()
        p.join()
    # report elapsed time
```

```
    elapsed_time = time.time() - start_time
    elapsed_time = f'{elapsed_time:>12.3f} seconds\n'
    sys.stderr.write("Elapsed time: " + elapsed_time)
```

## Python3-App-E. Parallel mode: imap

```
$ cat run_queries_imap.py
#!/usr/local/bin/python3
""" run_queries_imap.py """
# pylint: disable=invalid-name, import-error, line-too-long
import multiprocessing as mp
import os
from pathlib import Path
from signal import signal, SIGINT
import sys
# pylint: disable=W0622
from sys import exit, stdin
import time
from time import strftime, gmtime
# https://github.com/TeskaLabs/cysimdjson
import cysimdjson
# https://requests.readthedocs.io/en/latest/
import requests
# DNSDB API allows up to a maximum of 10 concurrent query streams
processes = 10
def get_api_key():
    """ Retrieve the DNSDB API key """
    api_key_file_path = os.path.join(str(Path.home()),
".dnsdb-apikey.txt")
    try:
        with open(api_key_file_path, encoding='UTF-8') as my_api_file:
            val = my_api_file.readline().strip()
    except FileNotFoundError:
        exit("DNSDB API key should be in ~/.dnsdb-apikey.txt")
    return val
def handler(_ , _2):
    """Stub routine to handle the user hitting ctrl-C"""
    exit("\nUser hit ctrl-C -- exiting")
def make_query(myfqdn):
    """ Make the DNSDB query for myfqdn """
    # global variables aren't passed to spawned processes, so we can't
just
```

```
    # get the API key once-and-done
    # get the DNSDB API key
    my_apikey = get_api_key()
    url = "https://api-pao.dnsdb.info/dnsdb/v2/lookup/rrset/name/"+myfqdn+
\

"?limit=1000000&time_last_after=1668377911&time_first_before=1670969911"
    myheaders = {'X-API-Key': my_apikey, 'Accept': 'application/jsonl'}
    r = requests.get(url, headers=myheaders, timeout=3600)
    # Status Code 200 == Success
    if r.status_code == 200:
        stripped_results = remove_saf_entries(r.text)
        for myfqdns in stripped_results:
            myline = print_detailed_bits(myfqdns)
            if myline is not None:
                sys.stdout.write(myline)
        sys.stdout.flush()
    else:
        sys.stderr.write(myfqdn+" returned status
code="+r.status_code+"\n")
def print_detailed_bits(myrecord):
    """format results for display"""
    parser = cysimdjson.JSONParser()
    myrecord_json_format = parser.loads(myrecord)
    my_rrtype = myrecord_json_format['obj']['rrtype']
    # assume we're only interested in "A" records and CNAMEs
    if my_rrtype.rstrip() not in ("A", "CNAME"):
        return None
    my_rrname = myrecord_json_format['obj']['rrname']
    my_count = myrecord_json_format['obj']['count']
    my_rdata = str((myrecord_json_format['obj']['rdata']).export())
    myformat = '%y-%m-%d %H:%M'
    # time_last
    try:
        extract_tl = myrecord_json_format['obj']['time_last']
    except KeyError:
        extract_tl = myrecord_json_format['obj']['zone_time_last']
    enddatetime = strftime(myformat, gmtime(extract_tl))
    # time_first
    try:
        extract_tf = myrecord_json_format['obj']['time_first']
    except KeyError:
        extract_tf = myrecord_json_format['obj']['zone_time_first']
    startdatetime = strftime(myformat, gmtime(extract_tf))
```

```
    # format the output for display
    my_rrname = f'{my_rrname:<55}'
    my_rrtype = f'{my_rrtype:<6}'
    my_count = f'{my_count:>12,}'
    # enquoting the date times requires that we use escaped quotes
    results = f'{my_rrname} {my_rrtype} \"{enddatetime}\"
\"{startdatetime}\" {my_count} {my_rdata}\n'
    return results
def remove_saf_entries(mylist):
    """ Remove the Streaming API Framing Records From the Results """
    mylist2 = mylist.splitlines()
    # Strip the streaming format bookend records
    if mylist2[0] == '{"cond":"begin"}':
        mylist2.pop(0)
    if ((mylist2[-1] == '{"cond":"succeeded"}') or \
        (mylist2[-1] == '{"cond":"limited","msg":"Result limit
reached"}')):
        mylist2.pop()
    return mylist2
if __name__ == "__main__":
    # Did you forget to pipe in the sites to check?
    if stdin.isatty():
        exit("Pipe in sites to check via stdin")
    # get timing start time
    start_time = time.time()
    # spawn is the default for MacOS, but fork is the default for many
other Un*x systems
    # if spawn's not used, fork has implications for global variable
inheritance, etc.
    mp.set_start_method('spawn')
    # handle ctrl-C interrupt cleanly
    signal(SIGINT, handler)
    fqdns = []
    for line in stdin:
        fqdns.append(line.strip())
    # process the sites (across the various processes)
    with mp.Pool(processes) as p:
        p.imap(make_query, fqdns, chunksize=8)
        p.close()
        p.join()
    # report elapsed time
    elapsed_time = time.time() - start_time
    elapsed_time = f'{elapsed_time:>12.3f} seconds\n'
    sys.stderr.write("Elapsed time: " + elapsed_time)
```

## Python3-App-F. Parallel mode: threaded version

```
$ cat run_queries_threaded_mode.py
#!/usr/local/bin/python3
""" run_queries_threaded_mode.py """
# pylint: disable=invalid-name, import-error, line-too-long
import multiprocessing.dummy as mp
import os
from pathlib import Path
from signal import signal, SIGINT
import sys
# pylint: disable=W0622
from sys import exit, stdin
import time
from time import strftime, gmtime
# https://github.com/TeskaLabs/cysimdjson
import cysimdjson
# https://requests.readthedocs.io/en/latest/
import requests
# DNSDB API allows up to a maximum of 10 concurrent query streams
threads = 10
def get_api_key():
    """ Retrieve the DNSDB API key """
    api_key_file_path = os.path.join(str(Path.home()),
".dnsdb-apikey.txt")
    try:
        with open(api_key_file_path, encoding='UTF-8') as my_api_file:
            val = my_api_file.readline().strip()
    except FileNotFoundError:
        exit("DNSDB API key should be in ~/.dnsdb-apikey.txt")
    return val
def handler(_ , _2):
    """Stub routine to handle the user hitting ctrl-C"""
    exit("\nUser hit ctrl-C -- exiting")
def make_query(myfqdn):
    """ Make the DNSDB query for myfqdn """
    # get the DNSDB API key
    my_apikey = get_api_key()
    url = "https://api-pao.dnsdb.info/dnsdb/v2/lookup/rrset/name/"+myfqdn+
\
```

```
"?limit=1000000&time_last_after=1668377911&time_first_before=1670969911"
    myheaders = {'X-API-Key': my_apikey, 'Accept': 'application/jsonl'}
    r = requests.get(url, headers=myheaders, timeout=3600)
    # Status Code 200 == Success
    if r.status_code == 200:
        stripped_results = remove_saf_entries(r.text)
        for myfqdns in stripped_results:
            myline = print_detailed_bits(myfqdns)
            if myline is not None:
                sys.stdout.write(myline)
        sys.stdout.flush()
    else:
        sys.stderr.write(myfqdn+" returned status
code="+r.status_code+"\n")
def print_detailed_bits(myrecord):
    """format results for display"""
    parser = cysimdjson.JSONParser()
    myrecord_json_format = parser.loads(myrecord)
    my_rrtype = myrecord_json_format['obj']['rrtype']
    # assume we're only interested in "A" records and CNAMEs
    if my_rrtype.rstrip() not in ("A", "CNAME"):
        return None
    my_rrname = myrecord_json_format['obj']['rrname']
    my_count = myrecord_json_format['obj']['count']
    my_rdata = str((myrecord_json_format['obj']['rdata']).export())
    myformat = '%y-%m-%d %H:%M'
    # time_last
    try:
        extract_tl = myrecord_json_format['obj']['time_last']
    except KeyError:
        extract_tl = myrecord_json_format['obj']['zone_time_last']
    enddatetime = strftime(myformat, gmtime(extract_tl))
    # time_first
    try:
        extract_tf = myrecord_json_format['obj']['time_first']
    except KeyError:
        extract_tf = myrecord_json_format['obj']['zone_time_first']
    startdatetime = strftime(myformat, gmtime(extract_tf))
    # format the output for display
    my_rrname = f'{my_rrname:<55}'
    my_rrtype = f'{my_rrtype:<6}'
    my_count = f'{my_count:>12,}'
    # enquoting the date times requires that we use escaped quotes
```

```
    results = f'{my_rrname} {my_rrtype} \"{enddatetime}\"
\"{startdatetime}\" {my_count} {my_rdata}\n'
    return results
def remove_saf_entries(mylist):
    """ Remove the Streaming API Framing Records From the Results """
    mylist2 = mylist.splitlines()
    # Strip the streaming format bookend records
    if mylist2[0] == '{"cond":"begin"}':
        mylist2.pop(0)
    if ((mylist2[-1] == '{"cond":"succeeded"}') or \
        (mylist2[-1] == '{"cond":"limited","msg":"Result limit
reached"}')):
        mylist2.pop()
    return mylist2
if __name__ == "__main__":
    # Did you forget to pipe in the sites to check?
    if stdin.isatty():
        exit("Pipe in sites to check via stdin")
    # get timing start time
    start_time = time.time()
    # handle ctrl-C interrupt cleanly
    signal(SIGINT, handler)
    fqdns = []
    for line in stdin:
        fqdns.append(line.strip())
    # process the sites (across the various threads)
    with mp.Pool(threads) as p:
        p.imap_unordered(make_query, fqdns, chunksize=8)
        p.close()
        p.join()
    # report elapsed time
    elapsed_time = time.time() - start_time
    elapsed_time = f'{elapsed_time:>12.3f} seconds\n'
    sys.stderr.write("Elapsed time: " + elapsed_time)
```

## Python3-App-G. Parallel mode: asyncio version

```
$ cat run_asyncio.py
#!/usr/local/bin/python3
""" run_asyncio.py """
# pylint: disable=invalid-name, import-error, line-too-long
import os
```

```
from pathlib import Path
import resource
import sys
# pylint: disable=W0622
from sys import exit, stdin
import time
from time import strftime, gmtime
# https://docs.python.org/3/library/asyncio.html
import asyncio
# https://docs.aiohttp.org/en/stable/
import aiohttp
# https://github.com/TeskaLabs/cysimdjson
import cysimdjson
# Default open files may be too low on macOS:
https://github.com/saghul/aiodns/issues/50
# Fix this at the *** shell prompt ***: ulimit -S -n 50000
new_soft_limit = 50000
(rlimit_nofile_soft, rlimit_nofile_hard) =
resource.getrlimit(resource.RLIMIT_NOFILE)
resource.setrlimit(resource.RLIMIT_NOFILE, (new_soft_limit,
rlimit_nofile_hard))
# DNSDB API allows up to 10 concurrent query streams
max_sessions=10
def get_api_key():
    """ Retrieve the DNSDB API key """
    api_key_file_path = os.path.join(str(Path.home()),
".dnsdb-apikey.txt")
    try:
        with open(api_key_file_path, encoding='UTF-8') as my_api_file:
            val = my_api_file.readline().strip()
    except FileNotFoundError:
        exit("DNSDB API key should be in ~/.dnsdb-apikey.txt")
    return val
async def make_query(myfqdn, semaphore) -> bytes:
    """ Make the DNSDB query for myfqdn """
    # get the DNSDB API key
    my_apikey = get_api_key()
    myheaders = {'X-API-Key': my_apikey, 'Accept': 'application/jsonl'}
    url = "https://api-pao.dnsdb.info/dnsdb/v2/lookup/rrset/name/"+myfqdn+
\

"?limit=1000000&time_last_after=1668377911&time_first_before=1670969911"
    # make an aiohttp call to DNSDB (we're using this instead of
"requests")
```

```
    async with aiohttp.ClientSession(headers=myheaders) as session:
        await semaphore.acquire()
        async with session.get(url, timeout=3600) as resp:
            content = await resp.read()
            semaphore.release()
            return content
async def write_to_stdout(content: bytes) -> None:
    """ return the results asynchronously """
    # convert from bytes to string
    content2 = content.decode()
    stripped_results = remove_saf_entries(content2)
    for mylines in stripped_results:
        oneline = print_detailed_bits(mylines)
        if oneline is not None:
            sys.stdout.write(oneline)
    sys.stdout.flush()
async def get_and_dump_dnsdb_results(myfqdn, semaphore) -> None:
    """ wrapper to combine the async dnsdb query and async results output
"""
    content = await make_query(myfqdn, semaphore)
    await write_to_stdout(content)
def print_detailed_bits(myrecord):
    """format results for display"""
    parser = cysimdjson.JSONParser()
    myrecord_json_format =  parser.loads(myrecord)
    my_rrtype = myrecord_json_format['obj']['rrtype']
    # assume we're only interested in "A" records and CNAMEs
    if my_rrtype.rstrip() not in ("A", "CNAME"):
        return None
    my_rrname = myrecord_json_format['obj']['rrname']
    my_rdata = str((myrecord_json_format['obj']['rdata']).export())
    my_count = myrecord_json_format['obj']['count']
    myformat = '%y-%m-%d %H:%M'
    # time_last
    try:
        extract_tl = myrecord_json_format['obj']['time_last']
    except KeyError:
        extract_tl = myrecord_json_format['obj']['zone_time_last']
    enddatetime = strftime(myformat, gmtime(extract_tl))
    # time_first
    try:
        extract_tf = myrecord_json_format['obj']['time_first']
    except KeyError:
        extract_tf = myrecord_json_format['obj']['zone_time_first']
```

```
    startdatetime = strftime(myformat, gmtime(extract_tf))
    # format the output for display
    my_rrname = f'{my_rrname:<55}'
    my_rrtype = f'{my_rrtype:<6}'
    my_count = f'{my_count:>12,}'
    # enquoting the date times requires that we use escaped quotes
    results = f'{my_rrname} {my_rrtype} \"{enddatetime}\"
\"{startdatetime}\" {my_count} {my_rdata}\n'
    return results
def remove_saf_entries(mylist):
    """ Remove the Streaming API Framing Records From the Results """
    mylist2 = mylist.splitlines()
    # Strip the streaming format bookend records
    if mylist2[0] == '{"cond":"begin"}':
        mylist2.pop(0)
    if ((mylist2[-1] == '{"cond":"succeeded"}') or \
        (mylist2[-1] == '{"cond":"limited","msg":"Result limit
reached"}')):
        mylist2.pop()
    return mylist2
async def main():
    """ main async routine """
    semaphore = asyncio.Semaphore(value=max_sessions)
    lines = []
    for line in stdin:
        lines.append(get_and_dump_dnsdb_results(line.strip(), semaphore))
    await asyncio.gather(*lines)
if __name__ == "__main__":
    # Did you forget to pipe in the sites to check?
    if stdin.isatty():
        exit("Pipe in sites to check via stdin")
    # get timing start time
    start_time = time.time()
    loop = asyncio.new_event_loop()
    asyncio.set_event_loop(loop)
    loop.run_until_complete(main())
    loop.close()
    # report elapsed time
    elapsed_time = time.time() - start_time
    elapsed_time = f'{elapsed_time:>12.3f} seconds\n'
    sys.stderr.write("Elapsed time: " + elapsed_time)
```

# "R" CODE APPENDICIES

> *"I just wish that Statistics was as easy as arranging numbers in chronological order, finding the median, lower and upper quartiles, and placing them on a box and whisker chart."*
> - *Charmaine J. Forde*

## R-App-A. Sequential Mode Analysis

```
$ cat univariate-serial-python.R
# these run_time values are for the serial Python code
run_time <-     c(4250.735, 4555.443, 4989.667, 5112.302, 4607.801,
                  4429.962, 3502.503, 3306.125, 3450.785, 3405.594)
# these result_count values are for the serial Python code, too
result_count <- c(20815193, 20815631, 20816124, 20824176, 20824702,
                  20825376, 20825779, 20826240, 20826723, 20827005)
x <- seq(1, length(run_time))
length(run_time)
mean(run_time)
sd(run_time)
min(run_time)
median(run_time)
max(run_time)
max(run_time) - min(run_time)
run_time_model <- lm(run_time ~ x)
summary(run_time_model)
pdf(file = "serial_python_dot_plot_run_time.pdf")
plot(x, run_time, col = "black", ylab = "seconds")
abline(run_time_model)
title(main = "Univariate RUN TIMES")
pdf(file = "serial_python_boxplot_run_time.pdf")
boxplot(run_time, ylab = "seconds", name = "Run Time", xlab = "")
title(main = "Univariate RUN TIMES")
length(result_count)
```

```
mean(result_count)
sd(result_count)
min(result_count)
median(result_count)
max(result_count)
max(result_count) - min(result_count)
result_count_model <- lm(result_count ~ x)
summary(result_count_model)
pdf(file = "serial_python_dot_plot_result_count.pdf")
plot(x, result_count, col = "black")
abline(result_count_model)
title(main = "Univariate RESULT COUNTS")
pdf(file = "serial_python_boxplot_result_count.pdf")
boxplot(result_count, ylab = "# of results", name = "Results")
title(main = "Univariate RESULT COUNTS")
```

## R-App-B: Parallel mode: imap.unordered

```
$ cat univariate-parallel-1.R
# these run_time values are for the imap-unordered parallel Python code
run_time <-       c(481.536, 476.182, 452.413, 447.027, 449.576,
                    454.866, 458.208, 445.533, 517.817, 470.027)
# these result_count values are for the imap-unordered parallel Python
code, too
result_count <-   c(20836814, 20837352, 20837569, 20837701, 20837810,
                    20837894, 20838004, 20838103, 20838390, 20838672)
x <- seq(1, length(run_time))
length(run_time)
mean(run_time)
sd(run_time)
min(run_time)
median(run_time)
max(run_time)
max(run_time) - min(run_time)
pdf(file = "parallel_I_python_dot_plot_run_time.pdf")
plot(x, run_time, col = "black", pch = 16, xlab = "")
title(main = "Parallel imap.unordered (chunksize=8) RUN TIMES")
pdf(file = "parallel_I_python_boxplot_run_time.pdf")
boxplot(run_time, ylab = "seconds", names = "Run Time")
title(main = "Parallel imap.unordered (chunksize=8) RUN TIMES")
length(result_count)
mean(result_count)
```

```
sd(result_count)
min(result_count)
median(result_count)
max(result_count)
max(result_count) - min(result_count)
pdf(file = "parallel_I_python_dot_plot_result_count.pdf")
plot(x, result_count, col = "red", pch = 16, xlab = "")
title(main = "Parallel imap.unordered (chunksize=8) RESULT COUNTS")
pdf(file = "parallel_I_python_boxplot_result_count.pdf")
boxplot(result_count, ylab = "# of results", name = "Results")
title(main = "Parallel imap.unordered (chunksize=8) RESULT COUNTS")
```

## R-App-C. Parallel mode: map

```
$ cat univariate-parallel-map.R
# these run_time values are for the plain map parallel Python code
# longer timeout, designated node, fixed time fences
run_time <-       c(443.553, 448.175, 444.279, 469.960, 451.423,
                    432.002, 392.891, 423.294, 387.184, 377.515)
# these result_count values are for the plain map parallel Python code,
too
# longer timeout, designated node, fixed time fences
result_count <-   c(20839787, 20839915, 20840083, 20840258, 20840405,
                    20840650, 20840827, 20840958, 20841047, 20841297)
x <- seq(1, length(run_time))
length(run_time)
mean(run_time)
sd(run_time)
min(run_time)
median(run_time)
max(run_time)
max(run_time) - min(run_time)
pdf(file = "parallel_I_map_python_dot_plot_run_time.pdf")
plot(x, run_time, col = "black", pch = 16, xlab = "")
title(main = "Parallel map RUN TIMES")
pdf(file = "parallel_I_map_python_boxplot_run_time.pdf")
boxplot(run_time, ylab = "seconds", names = "Run Time")
title(main = "Parallel map RUN TIMES")
length(result_count)
mean(result_count)
sd(result_count)
min(result_count)
```

```
median(result_count)
max(result_count)
max(result_count) - min(result_count)
pdf(file = "parallel_I_map_python_dot_plot_result_count.pdf")
plot(x, result_count, col = "red", pch = 16, xlab = "")
title(main = "Parallel map RESULT COUNTS")
pdf(file = "parallel_I_map_python_boxplot_result_count.pdf")
boxplot(result_count, ylab = "# of results", name = "Results")
title(main = "Parallel map RESULT COUNTS")
```

## R-App-D. Parallel mode: imap

```
$ cat univariate-parallel-imap.R
# these run_time values are for the imap parallel Python code
run_time <-     c(436.497, 428.283, 378.741, 391.764, 386.347,
                  382.980, 382.547, 385.509, 385.960, 376.456)
# these result_count values are for the imap parallel Python code, too
result_count <- c(20857131, 20857220, 20857343, 20857432, 20857498,
                  20857832, 20858160, 20858231, 20858312, 20858380)
x <- seq(1, length(run_time))
length(run_time)
mean(run_time)
sd(run_time)
min(run_time)
median(run_time)
max(run_time)
max(run_time) - min(run_time)
pdf(file = "parallel_I_imap_python_dot_plot_run_time.pdf")
plot(x, run_time, col = "black", pch = 16, xlab = "")
title(main = "Parallel imap chunksize=8 RUN TIMES")
pdf(file = "parallel_I_imap_python_boxplot_run_time.pdf")
boxplot(run_time, ylab = "seconds", names = "Run Time")
title(main = "Parallel imap chunksize=8 RUN TIMES")
length(result_count)
mean(result_count)
sd(result_count)
min(result_count)
median(result_count)
max(result_count)
max(result_count) - min(result_count)
pdf(file = "parallel_I_imap_python_dot_plot_result_count.pdf")
plot(x, result_count, col = "red", pch = 16, xlab = "")
```

```
title(main = "Parallel imap chunksize=8 RESULT COUNTS")
pdf(file = "parallel_I_imap_python_boxplot_result_count.pdf")
boxplot(result_count, ylab = "# of results", name = "Results")
title(main = "Parallel imap chunksize=8 RESULT COUNTS")
```

## R-App-E. Parallel mode: threaded version

```
$ cat univariate-threaded.R
# these run_time values are for the threaded parallel Python code
run_time <-     c(428.203, 434.683, 432.897, 446.154, 435.250,
                  432.910, 456.082, 451.791, 464.881, 470.497)
# these result_count values are for the threaded parallel Python code, too
result_count <- c(20859436, 20859558, 20859750, 20859814, 20861034,
                  20861282, 20861405, 20861482, 20861648, 20861905)
x <- seq(1, length(run_time))
mean(run_time)
sd(run_time)
min(run_time)
median(run_time)
max(run_time)
max(run_time) - min(run_time)
pdf(file = "threaded_dot_plot_run_time.pdf")
plot(x, run_time, col = "black", pch = 16, xlab = "")
title(main = "Threaded RUN TIMES")
pdf(file = "threaded_boxplot_run_time.pdf")
boxplot(run_time, ylab = "seconds", names = "Run Time")
title(main = "Threaded RUN TIMES")
mean(result_count)
sd(result_count)
min(result_count)
median(result_count)
max(result_count)
max(result_count) - min(result_count)
pdf(file = "threaded_dot_plot_result_count.pdf")
plot(x, result_count, col = "red", pch = 16, xlab = "")
title(main = "Threaded RESULT COUNTS")
pdf(file = "threaded_boxplot_result_count.pdf")
boxplot(result_count, ylab = "# of results", name = "Results")
title(main = "Threaded RESULT COUNTS")
```

## R-App-F. Parallel mode: asyncio version

```
$ cat univariate-parallel-asyncio.R
# these run_time values are for the asyncio parallel Python code
# longer timeout, designated node, fixed time fences
run_time <-       c(454.811, 445.150, 432.975, 440.346, 436.196,
                    455.818, 436.662, 444.055, 475.578, 480.435)
# these result_count values are for the asyncio parallel Python code, too
# longer timeout, designated node, fixed time fences
result_count <-   c(20891690, 20892773, 20893705, 20894425, 20895066,
                    20895787, 20896658, 20897005, 20898307, 20898802)
x <- seq(1, length(run_time))
length(run_time)
mean(run_time)
sd(run_time)
min(run_time)
median(run_time)
max(run_time)
max(run_time) - min(run_time)
pdf(file = "parallel_II_asyncio_python_dot_plot_run_time.pdf")
plot(x, run_time, col = "black", pch = 16, xlab = "")
title(main = "Parallel asyncio RUN TIMES")
pdf(file = "parallel_II_asyncio_python_boxplot_run_time.pdf")
boxplot(run_time, ylab = "seconds", names = "Run Time")
title(main = "Parallel asyncio RUN TIMES")
length(result_count)
mean(result_count)
sd(result_count)
min(result_count)
median(result_count)
max(result_count)
max(result_count) - min(result_count)
pdf(file = "parallel_II_asyncio_python_dot_plot_result_count.pdf")
plot(x, result_count, col = "red", pch = 16, xlab = "")
title(main = "Parallel asyncio RESULT COUNTS")
pdf(file = "parallel_II_asyncio_python_boxplot_result_count.pdf")
boxplot(result_count, ylab = "# of results", name = "Results")
title(main = "Parallel asyncio RESULT COUNTS")
```

## R-App-G. Combined Run

```
$ cat cross-group.R
# serial Python code
run_time_serial     <- c(4250.735, 4555.443, 4989.667, 5112.302, 4607.801,
                         4429.962, 3502.503, 3306.125, 3450.785, 3405.594)
count_serial        <- c(20815193, 20815631, 20816124, 20824176, 20824702,
                         20825376, 20825779, 20826240, 20826723, 20827005)
# imap-unordered parallel Python code
run_time_imap_unord <- c(481.536, 476.182, 452.413, 447.027, 449.576,
                         454.866, 458.208, 445.533, 517.817, 470.027)
count_imap_unord    <- c(20836814, 20837352, 20837569, 20837701, 20837810,
                         20837894, 20838004, 20838103, 20838390, 20838672)
# map parallel Python code
run_time_map <- c(443.553, 448.175, 444.279, 469.960, 451.423,
                  432.002, 392.891, 423.294, 387.184, 377.515)
count_map    <- c(20839787, 20839915, 20840083, 20840258, 20840405,
                  20840650, 20840827, 20840958, 20841047, 20841297)
# imap parallel Python code
run_time_imap <- c(436.497, 428.283, 378.741, 391.764, 386.347,
                   382.980, 382.547, 385.509, 385.960, 376.456)
count_imap    <- c(20857131, 20857220, 20857343, 20857432, 20857498,
                   20857832, 20858160, 20858231, 20858312, 20858380)
# thread parallel Python code
run_time_thread <- c(428.203, 434.683, 432.897, 446.154, 435.250,
                     432.910, 456.082, 451.791, 464.881, 470.497)
count_thread    <- c(20859436, 20859558, 20859750, 20859814, 20861034,
                     20861282, 20861405, 20861482, 20861648, 20861905)
# async parallel Python code
run_time_asyncio <- c(454.811, 445.150, 432.975, 440.346, 436.196,
                      455.818, 436.662, 444.055, 475.578, 480.435)
count_asyncio    <- c(20891690, 20892773, 20893705, 20894425, 20895066,
                      20895787, 20896658, 20897005, 20898307, 20898802)
x1 <- seq(1,  10)
x2 <- seq(11, 20)
x3 <- seq(21, 30)
x4 <- seq(31, 40)
x5 <- seq(41, 50)
x6 <- seq(51, 60)
time_x <- seq(1, 60)
time_y <- c(run_time_serial, run_time_imap_unord,
            run_time_map, run_time_imap,
            run_time_thread, run_time_asyncio)
```

```
my_run_time <- data.frame(run_time_serial, run_time_imap_unord,
                          run_time_map, run_time_imap,
                          run_time_thread, run_time_asyncio)
counts_y <- c(count_serial, count_imap_unord,
              count_map, count_imap,
              count_thread, count_asyncio)
my_counts <- data.frame(count_serial, count_imap_unord,
                        count_map, count_imap,
                        count_thread, count_asyncio)
my_groups <- c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
               3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
               4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
               5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
               6, 6, 6, 6, 6, 6, 6, 6, 6, 6)
my_chars <- c(16, 16, 16, 16, 16, 16, 16, 16, 16, 16,
               8,  8,  8,  8,  8,  8,  8,  8,  8,  8,
              15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
               1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
               4,  4,  4,  4,  4,  4,  4,  4,  4,  4,
              12, 12, 12, 12, 12, 12, 12, 12, 12, 12)
pdf(file = "combined_dot_plots_run_time.pdf")
plot(time_x, time_y, pch = my_chars, xlab = "", ylab = "seconds", xaxt =
"n")
title(main = "RUN TIMES in Seconds")
legend("topright", inset = 0.02,
       legend = c("Serial",
                  "Parallel imap_unordered",
                  "Parallel map",
                  "Parallel imap",
                  "Parallel thread",
                  "Parallel asyncio"),
       pch = c(16, 8, 15, 1, 4, 12))
pdf(file = "combined_boxplot_run_time.pdf")
boxplot(time_y ~ my_groups, ylab = "seconds")
title(main = "RUN TIMES in Seconds")
legend("topright", inset = 0.02,
   legend = c("1=Serial",
              "2=Parallel imap_unordered",
              "3=Parallel map",
              "4=Parallel imap",
              "5=Parallel thread",
              "6=Parallel asyncio"))
pdf(file = "combined_dot_plots_result_count.pdf")
```

```
plot(time_x, counts_y, pch = my_chars, xlab = "",
   ylab = "result count", xaxt = "n")
title(main = "RESULT COUNTS")
legend("topleft", inset = 0.02,
   legend = c("Serial",
              "Parallel imap_unordered",
              "Parallel map",
              "Parallel imap",
              "Parallel thread",
              "Parallel asyncio"),
   pch = c(16, 8, 15, 1, 4, 12))
pdf(file = "combined_boxplot_result_count.pdf")
boxplot(counts_y ~ my_groups, ylab = "result count")
title(main = "RESULT COUNTS")
legend("topleft", inset = 0.02,
   legend = c("1=Serial",
              "2=Parallel imap_unordered",
              "3=Parallel map",
              "4=Parallel imap",
              "5=Parallel thread",
              "6=Parallel asyncio"))
```

## R-App-H. Cythonized Asyncio

```
$ cat univariate-parallel-asyncio-cythonized.R
# these run_time values are for the asyncio parallel Python code
# CYTHONIZED
run_time <-       c(489.227, 547.059, 518.502, 508.796, 506.089,
                    509.494, 505.359, 484.640, 480.389, 475.454)
# these result_count values are for the asyncio parallel Python code, too
# CYTHONIZED
result_count <-   c(20942734, 20942807, 20942987, 20943033, 20943421,
                    20943510, 20943554, 20943638, 20943733, 20943791)
x <- seq(1, length(run_time))
length(run_time)
mean(run_time)
sd(run_time)
min(run_time)
median(run_time)
max(run_time)
max(run_time) - min(run_time)
run_time_model <- lm(run_time ~ x)
```

```
summary(run_time_model)
pdf(file = "cython_asyncio_python_dot_plot_run_time.pdf")
plot(x, run_time, col = "black", pch = 16, xlab = "")
abline(run_time_model)
title(main = "Parallel asyncio RUN TIMES -- CYTHONIZED")
pdf(file = "cython_asyncio_python_boxplot_run_time.pdf")
boxplot(run_time, ylab = "seconds", names = "Run Time")
title(main = "Parallel asyncio RUN TIMES -- CYTHONIZED")
length(result_count)
mean(result_count)
sd(result_count)
min(result_count)
median(result_count)
max(result_count)
max(result_count) - min(result_count)
result_count_model <- lm(result_count ~ x)
summary(result_count)
pdf(file = "cython_asyncio_python_dot_plot_result_count.pdf")
plot(x, result_count, col = "red", pch = 16, xlab = "")
abline(result_count_model)
title(main = "Parallel asyncio RESULT COUNTS -- CYTHONIZED")
pdf(file = "cython_asyncio_python_boxplot_result_count.pdf")
boxplot(result_count, ylab = "# of results", name = "Results")
title(main = "Parallel asyncio RESULT COUNTS -- CYTHONIZED")
```

## R-App-I. Asyncio, Focused on Result Count Growth

```
$ cat extra-univariate.R
# these run_time values are for the asyncio parallel Python code
# EXTENDED RUN
run_time <-       c(
494.724, 489.558, 463.698, 464.247, 468.632,
468.896, 443.843, 439.162, 445.292, 441.420,
440.123, 438.156, 450.133, 459.151, 440.813,
439.136, 440.537, 443.245, 444.290, 437.056,
448.297, 461.355, 454.866, 472.439, 448.358,
452.540, 467.788, 501.968, 497.852, 486.529,
491.009, 461.219, 438.467, 446.419, 452.363,
444.464, 481.820, 442.631, 442.093, 435.323,
429.542, 443.579, 447.674, 436.287, 428.700,
435.273, 428.949, 424.845, 439.465, 462.131,
462.765, 490.208, 510.300, 490.489, 469.429,
```

```
455.507, 505.514, 554.446, 576.782, 607.325,
581.390, 572.540, 553.665, 553.129, 559.977,
629.753, 588.155, 568.969, 559.297, 544.166,
498.862, 483.045, 486.408, 462.171, 451.717,
446.006, 451.675, 458.339, 467.501, 492.291,
482.797, 495.459, 475.572, 488.487, 486.803,
480.638, 524.126, 502.762, 533.019, 519.000,
517.137, 508.308, 515.540, 535.288, 530.555,
504.796, 523.388, 548.355, 524.516, 502.356,
508.369, 512.118, 491.081, 492.144, 484.199,
496.140, 491.729, 512.478, 503.518, 559.544,
551.764, 532.350, 529.548, 540.427, 565.331,
549.076, 527.844, 520.344, 527.150, 536.107,
537.495, 512.864, 492.670, 504.083, 454.799,
462.306, 462.866, 471.242, 472.230, 477.067,
462.225, 454.205, 459.213, 450.053, 447.454,
456.614, 478.199, 466.291, 460.455, 473.143,
457.692, 463.396, 449.820, 453.954, 450.660,
464.895, 459.406, 462.857, 458.847, 452.559,
458.084, 457.220, 448.324, 471.197, 449.365,
447.207, 463.982, 455.624, 459.705, 531.813,
521.102, 512.013, 492.684, 494.885, 504.137,
473.037, 522.529, 537.844, 529.816, 542.920,
525.005, 538.489, 545.290, 579.332, 557.655,
540.920, 537.602, 489.970, 484.459, 462.188,
439.206, 438.860, 431.998, 433.449, 428.041,
428.237, 428.529, 437.578, 428.203, 437.506,
434.032, 432.785, 447.409, 442.373, 441.851,
432.896, 454.096, 488.734, 498.572, 468.994)
# these result_count values are for the asyncio parallel Python code, too
# EXTENDED RUN
result_count <-   c(
20926576, 20926630, 20926663, 20926694, 20926721,
20926528, 20926898, 20926993, 20927037, 20927058,
20927075, 20927158, 20927202, 20927254, 20927322,
20927460, 20927595, 20927723, 20927780, 20927843,
20927898, 20928167, 20928238, 20928282, 20928326,
20928361, 20928422, 20928491, 20928601, 20928665,
20928747, 20928860, 20928903, 20928945, 20929081,
20929208, 20929316, 20929398, 20929543, 20929663,
20929712, 20929830, 20929895, 20929987, 20930086,
20930164, 20930364, 20930479, 20930598, 20930683,
20930793, 20930895, 20930975, 20931009, 20931039,
20931092, 20931170, 20931209, 20931304, 20931389,
```

```
20931546, 20931725, 20931823, 20932021, 20932184,
20932280, 20932391, 20932682, 20932935, 20933250,
20933286, 20933296, 20933388, 20933613, 20933748,
20933877, 20934010, 20934112, 20934287, 20934497,
20934683, 20934755, 20935064, 20935205, 20935308,
20935430, 20935557, 20935645, 20935687, 20935702,
20935740, 20935828, 20935862, 20935870, 20935939,
20936045, 20936077, 20936093, 20936175, 20936257,
20936293, 20936337, 20936363, 20936401, 20936430,
20936545, 20936621, 20936662, 20936694, 20936737,
20936772, 20936794, 20936834, 20936890, 20936937,
20936983, 20937006, 20937031, 20937090, 20937170,
20937201, 20937220, 20937319, 20937435, 20937462,
20937492, 20937573, 20937613, 20937629, 20937637,
20937639, 20937639, 20937639, 20937655, 20937780,
20937853, 20937968, 20938012, 20938033, 20938062,
20938088, 20938102, 20938135, 20938157, 20938166,
20938191, 20938235, 20938255, 20938291, 20938351,
20938373, 20938393, 20938417, 20938441, 20938478,
20938539, 20938566, 20938586, 20938603, 20938625,
20938709, 20938730, 20938949, 20939073, 20939138,
20939168, 20939420, 20939508, 20939648, 20939677,
20939714, 20939733, 20939751, 20939797, 20939940,
20939983, 20940019, 20940042, 20940089, 20940198,
20940359, 20940401, 20940477, 20940548, 20940659,
20940738, 20940864, 20940958, 20941033, 20941149,
20941202, 20941305, 20941467, 20941516, 20941588,
20942144, 20942234, 20942377, 20942450, 20942533)
x <- seq(1,length(run_time))
length(run_time)
mean(run_time)
sd(run_time)
min(run_time)
median(run_time)
max(run_time)
max(run_time) - min(run_time)
sum(run_time)
run_time_model <- lm(run_time ~ x)
summary(run_time_model)
pdf(file="extended_asyncio_python_dot_plot_run_time.pdf")
plot(x, run_time, col="black", pch=16)
abline(run_time_model)
title(main="asyncio EXTENDED REPITITIONS, RUN TIMES")
pdf(file="extended_asyncio_python_boxplot_run_time.pdf")
```

```
boxplot(run_time, ylab="seconds", names="Run Time")
title(main="asyncio EXTENDED REPITITIONS, RUN TIMES")
length(result_count)
mean(result_count)
sd(result_count)
min(result_count)
median(result_count)
max(result_count)
max(result_count) - min(result_count)
sum(result_count)
result_count_model <- lm(result_count ~ x)
summary(result_count_model)
pdf(file="extended_asyncio_python_dot_plot_result_count.pdf")
plot(x, result_count, col="red", pch=16)
abline(result_count_model)
title(main="asyncio EXTENDED REPETITIONS, RESULT COUNTS")
pdf(file="extended_asyncio_python_boxplot_result_count.pdf")
boxplot(result_count, ylab="# of results", name="Results")
title(main="ayncio EXTENDED REPETITIONS, RESULT COUNTS")
```

# MISCELLANEOUS APPENDICES

> *"Above all else, show the data."*
> *- Edward R. Tufte*

# Misc-App-1. Dot Edu Domains

```
$ cat all-edus.txt
*.22cf.edu
*.290tech.edu
*.3ponts.edu
*.3sheep.edu
*.4cd.edu
*.4dcollege.edu
*.aa.edu
*.aaa.edu
*.aaaom.edu
*.aaart.edu
*.aacc.edu
*.aacsb.edu
*.aada.edu
*.aae.edu
*.aahb.edu
*.aahctn.edu
*.aai.edu
*.aaimt.edu
*.aakers.edu
*.aami.edu
*.aamu.edu
*.aapt.edu
*.aast.edu
*.aati.edu
*.aationline.edu
*.aau.edu
*.aauj.edu
*.aauni.edu
*.aaup.edu
*.aaureg.edu
*.ab.edu
*.aba.edu
*.abac.edu
*.abalanguage.edu
*.abatoliba.edu
*.abbc.edu
*.abc.edu
*.abcnash.edu
*.abcollege.edu
*.abconline.edu
*.abcott.edu
*.abcs.edu
*.abctx.edu
*.abi.edu
*.abs.edu
*.abscosmo.edu
*.absw.edu
*.abt.edu
*.abtech.edu
*.abtu.edu
*.abu.edu
*.ac.edu
*.aca.edu
*.acacia.edu
*.academia.edu
*.academy.edu
*.academyart.edu
*.acadiana.edu
*.acaom.edu
*.acaydia.edu
*.acb.edu
*.acba.edu
*.acbhd.edu
*.acbrdu.edu
*.acbsp.edu
*.acc.edu
*.acc1.edu
*.access.edu
*.acchs.edu
*.acck.edu
*.accs.edu
*.acct.edu
*.acd.edu
*.acdireland.edu
*.ace.edu
*.acecareer.edu
*.acenet.edu
*.aces.edu
*.acg.edu
*.ach.edu
*.ache.edu
*.achehealth.edu
*.achs.edu
*.aci-train.edu
*.aci.edu
*.acinow.edu
*.acjvs.edu
*.acm.edu
*.acmc.edu
*.acmin.edu
*.acnt.edu
*.acom.edu
*.acon.edu
*.acot.edu
*.acp.edu
*.acpe.edu
*.acphs.edu
*.acr.edu
*.acs.edu
*.acsouth.edu
*.act-sf.edu
*.act.edu
*.actcm.edu
*.actcollege.edu
*.actraining.edu
*.actx.edu
*.acu.edu
*.acuonline.edu
*.acupuncture.edu
*.acusa.edu
*.acutabove.edu
*.adams.edu
*.adamsmith.edu
*.adamsu.edu
*.adc.edu
*.adec.edu
*.adelaide.edu
*.adelphi.edu
*.adhe.edu
*.adison.edu
*.adler.edu
*.admissons.edu
*.adrian.edu
*.adrians.edu
*.adschool.edu
*.adtec.edu
*.adu.edu
*.aduonline.edu
*.advanced.edu
*.aea.edu
*.aec.edu
*.aegean.edu
*.aeli.edu
*.aels.edu
*.aerobics.edu
*.aesa.edu
*.aesd.edu
*.aeu.edu
*.af.edu
*.afa.edu
*.afaa.edu
*.afi.edu
*.afit.edu
*.aflbs.edu
*.afms.edu
*.afna.edu
*.afoc.edu
*.africau.edu
*.afundacion.edu
*.agape.edu
*.agbu.edu
*.agmu.edu
*.agnes.edu
*.agnesscott.edu
*.ags.edu
*.agse.edu
*.agseminary.edu
*.agsm.edu
*.agts.edu
*.agu.edu
*.aha.edu
*.ahc.edu
*.ahci.edu
*.ahcp.edu
*.ahe.edu
*.ahec.edu
*.aherf.edu
*.ahgaff.edu
*.ahi.edu
*.ahlers.edu
*.ahos.edu
*.ahsu.edu
*.ahu.edu
*.ai.edu
*.aiam.edu
*.aib.edu
*.aibny.edu
*.aibonline.edu
*.aiboston.edu
*.aibschool.edu
*.aic-arts.edu
*.aic.edu
*.aica.edu
*.aicag.edu
*.aicampus.edu
*.aiccu.edu
*.aicm.edu
*.aicreative.edu
*.aicuo.edu
*.aicusa.edu
*.aid.edu
*.aids.edu
*.aidt.edu
*.aie.edu
*.aifl.edu
*.aiht.edu
*.aii.edu
*.aiias.edu
*.aiid.edu
*.aiims.edu
*.aiit.edu
*.aim.edu
*.aimc.edu
*.aimm.edu
*.aimqrezcrs.edu
*.aims.edu
*.aimschool.edu
*.aimsed.edu
*.aimt.edu
*.aiofhampton.edu
*.aioic.edu
*.aionline.edu
*.aiot.edu
*.aiou.edu
*.aipt.edu
*.aipx.edu
*.ais.edu
*.aish.edu
*.aism.edu
*.aiu.edu
*.aiub.edu
*.aiufl.edu
*.aiuniv.edu
*.aiuonline.edu
*.ajcunet.edu
*.ajr.edu
*.ajrca.edu
*.ajtraining.edu
*.aju.edu
*.ajula.edu
*.ak9i.edu
*.akbible.edu
*.aku.edu
*.ala.edu
*.alamancecc.edu
*.alameda.edu
*.alamo.edu
*.alanus.edu
*.alaska.edu
*.alaskacc.edu
*.alasu.edu
*.alb.edu
*.alba.edu
*.albany.edu
*.albanylaw.edu
*.albanytech.edu
*.albcschool.edu
*.albemarle.edu
*.albertus.edu
*.albion.edu
*.albizu.edu
*.albright.edu
*.alc.edu
*.alcorn.edu
*.aldergse.edu
*.alextech.edu
*.alfaisal.edu
*.alfonsiana.edu
*.alfred.edu
*.alfredadler.edu
*.alfredstate.edu
*.alfredtech.edu
*.alicelloyd.edu
*.alidallas.edu
*.all.edu
*.alleg.edu
*.allegany.edu
*.allegheny.edu
*.allencc.edu
*.allencol.edu
*.allenschool.edu
*.allhealth.edu
*.alliance.edu
*.allianceu.edu
*.alliant.edu
*.allied.edu
*.allison.edu
*.allstate.edu
*.allura.edu
*.alma.edu
*.almahdi.edu
*.alpenacc.edu
*.alphagroup.edu
*.alps.edu
*.alquds.edu
*.alsde.edu
*.altierus.edu
*.altoonactc.edu
*.alts.edu
*.alu.edu
*.alvernia.edu
*.alverno.edu
*.ama.edu
*.amat.edu
*.amazon.edu
*.amb.edu
*.ambassador.edu
*.ambassadors.edu
*.amberton.edu
*.amberu.edu
*.ambi.edu
*.ambria.edu
*.ambrose.edu
*.ambs.edu
*.amc.edu
*.amcats.edu
*.amcc.edu
*.amcd.edu
*.amcollege.edu
*.amda.edu
*.america.edu
*.american.edu
*.americana.edu
*.americare.edu
*.ameritech.edu
*.amg.edu
*.amherst.edu
*.amhscollege.edu
*.ami.edu
*.amiohio.edu
*.amity.edu
*.amiu.edu
*.amlotus.edu
*.amman.edu
*.amnc.edu
*.amnh.edu
*.amnu.edu
*.ampac.edu
*.ampi.edu
*.amrita.edu
*.amsamoa.edu
*.amsc.edu
*.amsu.edu
*.amt.edu
*.amtec.edu
*.amu.edu
*.an.edu
*.anaheim.edu
*.anamarc.edu
*.anc.edu
*.ancilla.edu
*.ancollege.edu
*.anderson.edu
*.andover.edu
*.andrews.edu
*.angelina.edu
*.angelo.edu
*.angley.edu
*.anho.edu
*.animschool.edu
*.annamaria.edu
*.annauniv.edu
*.annenberg.edu
*.annwebb.edu
*.anokaramsey.edu
*.anokatech.edu
*.anrcollege.edu
*.anselm.edu
*.antares.edu
*.anthem.edu
*.antigua.edu
*.antillespr.edu
*.antioch.edu
*.antiochla.edu
*.antiochne.edu
*.antiochsb.edu
*.antiochsea.edu
*.antonelli.edu
*.ants.edu
*.anu.edu
*.anyang.edu
*.aoa.edu
*.aohd.edu
*.aoicollege.edu
*.aoma.edu
*.aotearoa.edu
*.aou.edu
*.apb.edu
*.apc.edu
*.apeejay.edu
*.apexcvt.edu
*.apexsot.edu
*.api.edu
*.apiu.edu
*.apj.edu
*.apmi.edu
*.apnho.edu
*.apollo.edu
*.apollogrp.edu
*.apollos.edu
*.apostolic.edu
*.appa.edu
*.approachisc.edu
*.appstate.edu
*.aps.edu
*.apsu.edu
*.aptc.edu
*.apts.edu
*.apu.edu
*.apus.edu
*.aqsa.edu
*.aquinas.edu
*.arab.edu
*.arabia.edu
*.arapahoe.edu
*.arbor.edu
*.arbs.edu
*.arcadia.edu
*.archimede.edu
*.architect.edu
*.arclabs.edu
*.arellanolaw.edu
*.argosy.edu
*.arizona.edu
*.arkansas.edu
*.arknet.edu
*.arktech.edu
*.armstrong.edu
*.army.edu
*.arrl.edu
*.arsc.edu
*.art.edu
*.artacademy.edu
*.artcenter.edu
*.arti.edu
*.artic.edu
*.artistic.edu
*.aru.edu
*.as.edu
*.asa.edu
*.asaom.edu
*.asb.edu
*.asbury.edu
*.asc.edu
*.ascbs.edu
*.ascc.edu
*.ascenglish.edu
*.ascent.edu
*.asher.edu
*.ashford.edu
*.ashland.edu
*.ashmead.edu
*.asi.edu
*.asiapacific.edu
*.asicollege.edu
*.asismassage.edu
*.asl.edu
*.asm.edu
*.asn.edu
*.asnuntuck.edu
*.aspen.edu
*.aspenlaurel.edu
*.aspp.edu
*.asri.edu
*.assumption.edu
*.ast.edu
*.astate.edu
*.astrodome.edu
*.asu.edu
*.asub.edu
*.asumh.edu
*.asumidsouth.edu
*.asun.edu
*.asurams.edu
*.asusystem.edu
*.asutr.edu
*.ata.edu
*.atacollege.edu
*.atafl.edu
*.atc.edu
*.atcla.edu
*.atcsd.edu
*.atech.edu
*.ateneo.edu
*.atenet.edu
*.athabasca.edu
*.athena.edu
*.athenaeum.edu
*.athens.edu
*.athenstech.edu
*.ati.edu
*.aticollege.edu
*.atis.edu
*.atischools.edu
*.atitraining.edu
*.atlantatech.edu
*.atlantic.edu
*.atlanticu.edu
*.atlantis.edu
*.atlm.edu
*.atom.edu
```

*.atp.edu
*.atria.edu
*.ats.edu
*.atsu.edu
*.atu.edu
*.atyrau.edu
*.au-crib.edu
*.au.edu
*.aua.edu
*.aub.edu
*.aubg.edu
*.aubih.edu
*.aubs.edu
*.auburn.edu
*.auburnschl.edu
*.auc.edu
*.aucal.edu
*.aucegypt.edu
*.aucenter.edu
*.aucmed.edu
*.aucnyo.edu
*.auctr.edu
*.aud.edu
*.aug.edu
*.augie.edu
*.augsburg.edu
*.augusta.edu
*.augustana.edu
*.augustatech.edu
*.augustine.edu
*.auh.edu
*.auhs.edu
*.auhtc.edu
*.aui.edu
*.auis.edu
*.aul.edu
*.aum.edu
*.aup.edu
*.aupr.edu
*.aur.edu
*.auraria.edu
*.aurora.edu
*.aus.edu
*.aust.edu
*.austin.edu
*.austincc.edu
*.austingrad.edu
*.australia.edu
*.aut.edu
*.auto.edu
*.automeca.edu
*.autonoma.edu
*.autrytech.edu
*.avalon.edu
*.avc.edu
*.avcs.edu
*.aveda.edu
*.avedaarts.edu
*.avedafi.edu
*.avedanm.edu
*.avedapdx.edu
*.avedasouth.edu
*.avemaria.edu
*.avemarialaw.edu
*.aventis.edu
*.avenuefive.edu
*.averett.edu
*.aviation.edu
*.aviator.edu
*.avila.edu
*.avtec.edu
*.awbs.edu
*.awc.edu
*.awi.edu
*.ayers.edu
*.ayurveda.edu
*.ayurvedic.edu
*.azaruniv.edu
*.azculinary.edu
*.azregents.edu
*.azsummitlaw.edu
*.azure.edu
*.azusaadult.edu
*.azwestern.edu
*.b-sc.edu
*.babel.edu
*.babson.edu
*.bac.edu
*.bacone.edu
*.bae.edu
*.bai.edu
*.bainbridge.edu
*.baker.edu
*.bakerspgs.edu
*.bakeru.edu
*.balbharati.edu
*.bamasf.edu
*.bancroftsmt.edu
*.bankstreet.edu
*.baptist.edu
*.baptistu.edu
*.bar.edu
*.barbarigo.edu
*.bard.edu
*.barnard.edu
*.barpalma.edu
*.barrett.edu
*.barry.edu
*.barstow.edu
*.barton.edu
*.bartonccc.edu
*.baruch.edu
*.basf.edu
*.bassett.edu
*.bastyr.edu
*.batc.edu
*.bates.edu
*.batestech.edu
*.bath.edu
*.bau.edu
*.bauder.edu
*.bayantech.edu
*.baycollege.edu
*.baylor.edu
*.baypath.edu
*.bayside.edu
*.baystate.edu
*.bba.edu
*.bbc.edu
*.bbdnitm.edu
*.bbiny.edu
*.bbmi.edu
*.bbtc.edu
*.bbz.edu
*.bc.edu
*.bc3.edu
*.bca.edu
*.bcc.edu
*.bccc.edu
*.bccet.edu
*.bccr.edu
*.bce.edu
*.bchigh.edu
*.bchs.edu
*.bci.edu
*.bcit.edu
*.bcl.edu
*.bcm.edu
*.bcs.edu
*.bcsbc.edu
*.bcsmn.edu
*.bcu.edu
*.bcva.edu
*.bcw.edu
*.bd.edu
*.bds.edu
*.beach.edu
*.beacon.edu
*.beal.edu
*.bealcollege.edu
*.beaufortccc.edu
*.beaumont.edu
*.beauty.edu
*.beaver.edu
*.becbgk.edu
*.beck.edu
*.becker.edu
*.beckfield.edu
*.bedah-saraf.edu
*.bei.edu
*.belhaven.edu
*.bellarmine.edu
*.bellasa.edu
*.bellevue.edu
*.bellmar.edu
*.bellschool.edu
*.belmont.edu
*.beloit.edu
*.belrea.edu
*.bem.edu
*.ben.edu
*.benedict.edu
*.benedictine.edu
*.benes.edu
*.bennett.edu
*.bennington.edu
*.bentley.edu
*.berea.edu
*.berean.edu
*.bereanu.edu
*.bergen.edu
*.berginu.edu
*.berkeley.edu
*.berklee.edu
*.berkley-u.edu
*.berkley.edu
*.berkleyj.edu
*.berks.edu
*.berkshire.edu
*.berkshirecc.edu
*.berlitz.edu
*.bernelli.edu
*.berry.edu
*.betatech.edu
*.bethany-ca.edu
*.bethany.edu
*.bethanybc.edu
*.bethanygu.edu
*.bethanylb.edu
*.bethanywv.edu
*.bethbc.edu
*.bethel-in.edu
*.bethel.edu
*.bethelks.edu
*.bethelu.edu
*.bethesda.edu
*.bethlehem.edu
*.bethrivkah.edu
*.beulah.edu
*.bexley.edu
*.bfa.edu
*.bfcc.edu
*.bfit.edu
*.bfranklin.edu
*.bgh.edu
*.bgi.edu
*.bgsm.edu
*.bgsp.edu
*.bgsu.edu
*.bgu.edu
*.bhavuni.edu
*.bhc.edu
*.bhcarroll.edu
*.bhcc.edu
*.bhclr.edu
*.bhdi.edu
*.bhi.edu
*.bhmb.edu
*.bhs.edu
*.bhslr.edu
*.bhsu.edu
*.bhsy.edu
*.bhu.edu
*.bi.edu
*.bia.edu
*.bible.edu
*.bibleschool.edu
*.biblia.edu
*.biblical.edu
*.biblos.edu
*.bic.edu
*.bids.edu
*.bie.edu
*.bietdvg.edu
*.bigbend.edu
*.bilkent.edu
*.bim.edu
*.binghamton.edu
*.binus.edu
*.biola.edu
*.biop.edu
*.bios.edu
*.bircham.edu
*.birthingway.edu
*.birtraining.edu
*.birzeit.edu
*.bishop.edu
*.bishopbrady.edu
*.bit.edu
*.bju.edu
*.bklyn.edu
*.bkps.edu
*.blackburn.edu
*.blackhawk.edu
*.blackstone.edu
*.bladencc.edu
*.blaine.edu
*.blair.edu
*.blanquerna.edu
*.blc.edu
*.blinn.edu
*.bloomfield.edu
*.bloomu.edu
*.blts.edu
*.blue.edu
*.bluecc.edu
*.bluefield.edu
*.bluegrass.edu
*.blueridge.edu
*.bluewater.edu
*.bluffton.edu
*.bmats.edu
*.bmc.edu
*.bmcc.edu
*.bmg.edu
*.bmtc.edu
*.bnkst.edu
*.bnoschaim.edu
*.bodwell.edu
*.boisebible.edu
*.boisestate.edu
*.bonaventure.edu
*.bonn.edu
*.boscotech.edu
*.boston.edu
*.bottega.edu
*.bowdoin.edu
*.bowiestate.edu
*.bp.edu
*.bpc.edu
*.bpcc.edu
*.bpi.edu
*.bpkihs.edu
*.bradley.edu
*.braille.edu
*.bramsonort.edu
*.brandeis.edu
*.branden.edu
*.brandies.edu
*.brandman.edu
*.brasov.edu
*.braxton.edu
*.brazosport.edu
*.brc.edu
*.brcc.edu
*.brcn.edu
*.breining.edu
*.bremen.edu
*.brenau.edu
*.brensten.edu
*.brescia.edu
*.brevard.edu
*.brevardcc.edu
*.brewster.edu
*.briarcliff.edu
*.briarcliffe.edu
*.briarwood.edu
*.bricoh.edu
*.bridge.edu
*.bridgemont.edu
*.bridgeport.edu
*.bridges.edu
*.bridgew.edu
*.bridgewater.edu
*.briercrest.edu
*.brightpoint.edu
*.brightwood.edu
*.brioacademy.edu
*.bristol.edu
*.bristolcc.edu
*.brite.edu
*.broadview.edu
*.brockport.edu
*.broked.edu
*.brook.edu
*.brookdale.edu
*.brookdalecc.edu
*.brookes.edu
*.brookings.edu
*.brooklaw.edu
*.brookline.edu
*.brooklyn.edu
*.brooks.edu
*.brookstone.edu
*.brookwood.edu
*.broward.edu
*.brown.edu
*.brownaveda.edu
*.brownell.edu
*.browning.edu
*.brownmackie.edu
*.brownson.edu
*.brrice.edu
*.brsc.edu
*.brtech.edu
*.brunswickcc.edu
*.bryan.edu
*.bryant.edu
*.bryanu.edu
*.bryman.edu
*.brynathyn.edu
*.brynmawr.edu
*.bsa.edu
*.bsc.edu
*.bscc.edu
*.bse.edu
*.bshp.edu
*.bsk.edu
*.bsmcon.edu
*.bsom.edu
*.bsp.edu
*.bsstlearn.edu
*.bst.edu
*.bsu.edu
*.btc.edu
*.btech.edu
*.bths.edu
*.bti.edu
*.bts.edu
*.btsr.edu
*.bttc.edu
*.bu-spgs.edu
*.bu.edu
*.bua.edu
*.buc.edu
*.bucharest.edu
*.bucknell.edu
*.bucks.edu
*.buddhism.edu
*.buddhismus.edu
*.buffalo.edu
*.bumc.edu
*.burlington.edu
*.burnett.edu
*.burrell.edu
*.bush.edu
*.bushnell.edu
*.busm.edu
*.butc.edu
*.butler.edu
*.butlercc.edu
*.butlertech.edu
*.butte.edu
*.bvb.edu
*.bvinst.edu
*.bvu.edu
*.bw.edu
*.bxscience.edu
*.byts.edu
*.byu.edu
*.byuh.edu
*.byui.edu
*.byzcathsem.edu
*.c-tec.edu
*.ca.edu
*.cabrillo.edu
*.cabrini.edu
*.cac.edu
*.cacc.edu
*.caculinary.edu
*.caddokiowa.edu
*.cafe.edu
*.cahe.edu
*.cahsu.edu
*.cairn.edu
*.caj.edu
*.cala.edu
*.calaero.edu
*.calamerica.edu
*.calamuniv.edu
*.calarts.edu
*.calarttech.edu
*.calbapt.edu
*.calbaptist.edu
*.calc.edu
*.calcampus.edu
*.calcc.edu
*.calcoast.edu
*.caldwell.edu
*.calgary.edu
*.calhoun.edu
*.california.edu
*.calimt.edu
*.callutheran.edu
*.calmu.edu
*.calnorthern.edu
*.calpoly.edu
*.calsouthern.edu
*.calstate.edu
*.calstatela.edu
*.caltech.edu
*.calu.edu
*.calums.edu
*.calumsva.edu
*.calunion.edu
*.calvary.edu
*.calvin.edu
*.cambridge.edu
*.camdencc.edu
*.cameron.edu
*.campbell.edu
*.camphill.edu
*.campus.edu
*.cams.edu
*.camsen.edu
*.canberra.edu
*.cancerlinks.edu
*.canisius.edu
*.canton.edu
*.canyons.edu
*.capchrist.edu
*.cape.edu
*.capecod.edu
*.capella.edu
*.capital.edu
*.capitalcc.edu
*.capitol.edu
*.capri.edu
*.caprinow.edu
*.capstone.edu
*.captechu.edu
*.car.edu
*.cardean.edu
*.career.edu
*.careerquest.edu
*.careers.edu
*.careerta.edu
*.careertc.edu
*.careertech.edu
*.caribbean.edu
*.carlalbert.edu
*.carleton.edu
*.carlingford.edu
*.carlow.edu
*.carolina.edu
*.carolinau.edu
*.caroline.edu
*.carrington.edu
*.carroll.edu
*.carrollcc.edu
*.carrollu.edu
*.carsten.edu
*.carteret.edu
*.carthage.edu
*.carver.edu
*.cas.edu
*.casalaveda.edu
*.cascade.edu
*.cascadia.edu
*.case.edu
*.caspn.edu
*.castleton.edu
*.catawba.edu
*.catc.edu
*.catholic.edu
*.catlin.edu
*.catu.edu
*.cau.edu
*.cauniv.edu
*.cavt.edu
*.caycereilly.edu
*.cayuga-cc.edu
*.cazenovia.edu
*.cba.edu
*.cbc.edu
*.cbcag.edu
*.cbd.edu
*.cbet.edu
*.cbhs.edu
*.cbi.edu
*.cbnu.edu
*.cbp.edu
*.cbs.edu
*.cbseminary.edu
*.cbshouston.edu
*.cbt.edu
*.cbts.edu
*.cbu.edu
*.cbuonline.edu
*.cc-sd.edu
*.cc.edu
*.cca.edu
*.ccac-art.edu
*.ccac.edu
*.ccacademy.edu
*.ccad.edu
*.ccah.edu
*.ccal.edu
*.ccarts.edu
*.ccat.edu
*.ccaurora.edu
*.ccbatlanta.edu
*.ccbbc.edu
*.ccbc.edu
*.ccbcmd.edu
*.ccbcu.edu

*.ccbeu.edu
*.ccbs.edu
*.ccc.edu
*.cccb.edu
*.cccc.edu
*.cccco.edu
*.cccd.edu
*.cccneb.edu
*.cccnj.edu
*.cccoes.edu
*.cccollege.edu
*.cccs.edu
*.ccctc.edu
*.cccti.edu
*.cccua.edu
*.ccd.edu
*.ccdc.edu
*.cceionline.edu
*.ccg.edu
*.ccga.edu
*.cci.edu
*.ccicolleges.edu
*.ccinstitute.edu
*.ccis.edu
*.ccitraining.edu
*.cclemoyne.edu
*.ccls.edu
*.cclsmiami.edu
*.cclsnj.edu
*.ccm.edu
*.ccmcc.edu
*.ccmla.edu
*.ccms.edu
*.ccmt.edu
*.ccnh.edu
*.ccnm.edu
*.ccnn.edu
*.ccny.edu
*.ccon.edu
*.ccp.edu
*.ccpm.edu
*.ccr.edu
*.ccri.edu
*.ccs.edu
*.ccsd.edu
*.ccsdetroit.edu
*.ccsf.edu
*.ccsj.edu
*.ccsnh.edu
*.ccstudent.edu
*.ccsu.edu
*.cctc.edu
*.cctech.edu
*.ccu.edu
*.ccucollege.edu
*.ccv.edu
*.ccvc.edu
*.cda.edu
*.cdc.edu
*.cde.edu
*.cdi.edu
*.cdkc.edu
*.cdl.edu
*.cdrewu.edu
*.cds.edu
*.cdse.edu
*.cdsp.edu
*.cdu.edu
*.ceaprc.edu
*.cec.edu
*.cecil.edu
*.ceconline.edu
*.cedarcrest.edu
*.cedarville.edu
*.cedoc.edu
*.ceds.edu
*.ceea.edu
*.cei.edu
*.ceibs.edu
*.ceionline.edu
*.ceipi.edu
*.cel.edu
*.celebrity.edu
*.cemcollege.edu
*.cempr.edu
*.cen.edu
*.censolar.edu
*.centenary.edu
*.center.edu
*.centracon.edu
*.central.edu
*.centralaz.edu
*.centralia.edu
*.centraloc.edu
*.centralpenn.edu
*.centraltech.edu
*.centre.edu
*.centura.edu
*.century.edu
*.ceram.edu
*.cernet.edu
*.cerritos.edu
*.cerrocoso.edu
*.cersti.edu
*.ces.edu
*.cescollege.edu
*.cesi.edu
*.cesma.edu
*.cesna.edu
*.cet.edu
*.cetweb.edu
*.ceu.edu
*.cf.edu
*.cfbisd.edu
*.cfcc.edu
*.cffp.edu
*.cfi.edu
*.cfk.edu
*.cfot.edu
*.cga.edu
*.cgbbcaiup.edu
*.cgc.edu
*.cgcc.edu
*.cgi.edu
*.cgms.edu
*.cgs.edu
*.cgsc.edu
*.cgsot.edu
*.cgst.edu
*.cgu.edu
*.ch.edu
*.chabad.edu
*.chac.edu
*.chacu.edu
*.chadwick.edu
*.chafer.edu
*.chaffey.edu
*.chalcedon.edu
*.chamberlain.edu
*.chaminade.edu
*.champion.edu
*.champlain.edu
*.chancellor.edu
*.chancelloru.edu
*.chap-col.edu
*.chapin.edu
*.chapman.edu
*.charlotte.edu
*.charter.edu
*.charteroak.edu
*.chase.edu
*.chatfield.edu
*.chatham.edu
*.chattstate.edu
*.chc.edu
*.chcp.edu
*.chd.edu
*.chefs.edu
*.chemeketa.edu
*.cherylfell.edu
*.chesapeake.edu
*.chess.edu
*.cheveux.edu
*.cheyney.edu
*.chgosem.edu
*.chiangmai.edu
*.chic.edu
*.chicagogsb.edu
*.childmmc.edu
*.childsearch.edu
*.chipola.edu
*.chiu.edu
*.chivm.edu
*.chm.edu
*.choate.edu
*.chob.edu
*.choices.edu
*.chonnam.edu
*.chop.edu
*.chorro.edu
*.chowan.edu
*.chp.edu
*.christ.edu
*.christendom.edu
*.christies.edu
*.chrysm.edu
*.chsu.edu
*.chu.edu
*.chula.edu
*.chumsci.edu
*.chungang.edu
*.chw.edu
*.cia.edu
*.ciachef.edu
*.ciam.edu
*.cias.edu
*.ciat.edu
*.cibt.edu
*.cibu.edu
*.cic.edu
*.cicd99.edu
*.cid.edu
*.cide.edu
*.cidma.edu
*.cids.edu
*.cie-wc.edu
*.cie.edu
*.cif.edu
*.cihs.edu
*.ciis.edu
*.cile.edu
*.cim.edu
*.cims.edu
*.cimt.edu
*.cinec.edu
*.cintaaveda.edu
*.cioah.edu
*.cischools.edu
*.cisco.edu
*.cisl.edu
*.cisspat.edu
*.cit-liban.edu
*.cit.edu
*.citadel.edu
*.citcollege.edu
*.citrix.edu
*.cittx.edu
*.citycollege.edu
*.cityofhope.edu
*.cityu.edu
*.cityvision.edu
*.ciu.edu
*.ciula.edu
*.ciw.edu
*.ciwemb.edu
*.ciwt.edu
*.cjc.edu
*.cji.edu
*.ck.edu
*.ckp.edu
*.cks.edu
*.cktc.edu
*.cl.edu
*.cla.edu
*.clackamas.edu
*.claflin.edu
*.clajagxekl.edu
*.claremont.edu
*.clarion.edu
*.clarita.edu
*.clark.edu
*.clarkart.edu
*.clarke.edu
*.clarkson.edu
*.clarkstate.edu
*.clarku.edu
*.classmedia.edu
*.clatsopcc.edu
*.clayton.edu
*.clbi.edu
*.clc.edu
*.clcillinois.edu
*.clcmn.edu
*.cle-net.edu
*.cle.edu
*.clearpass.edu
*.clearwater.edu
*.cleary.edu
*.clemson.edu
*.cleveland.edu
*.clevelandcc.edu
*.clibc.edu
*.clic.edu
*.clinton.edu
*.cloud.edu
*.clovis.edu
*.clpccd.edu
*.cltc.edu
*.cltcc.edu
*.clu.edu
*.clunet.edu
*.cluniv.edu
*.cm.edu
*.cma.edu
*.cmc.edu
*.cmcc.edu
*.cmccd.edu
*.cmcnyls.edu
*.cmctx.edu
*.cmd.edu
*.cme.edu
*.cmh.edu
*.cmi.edu
*.cmich.edu
*.cmmccollege.edu
*.cmmcson.edu
*.cmn.edu
*.cmnok.edu
*.cmpr.edu
*.cmps.edu
*.cmqqxnomxn.edu
*.cms.edu
*.cmspath.edu
*.cmsv.edu
*.cmu.edu
*.cn.edu
*.cnc.edu
*.cncc.edu
*.cncusa.edu
*.cnd-md.edu
*.cnei.edu
*.cng.edu
*.cnicollege.edu
*.cnm.edu
*.cnr.edu
*.cnri.edu
*.cns.edu
*.cnsu.edu
*.cnu.edu
*.cnuas.edu
*.co-op.edu
*.coa.edu
*.coahomacc.edu
*.coastal.edu
*.coastalbend.edu
*.coastcareer.edu
*.coastline.edu
*.coba.edu
*.cobleskill.edu
*.coc.edu
*.cocc.edu
*.cochise.edu
*.coconino.edu
*.cod.edu
*.cods.edu
*.coe.edu
*.cofc.edu
*.coffeyville.edu
*.cofo.edu
*.cogr.edu
*.cogswell.edu
*.coker.edu
*.colby.edu
*.colbycc.edu
*.coleholland.edu
*.coleman.edu
*.colgate.edu
*.colin.edu
*.colleges.edu
*.collin.edu
*.collins-cc.edu
*.colmizpa.edu
*.colorado.edu
*.coloradomtn.edu
*.colostate.edu
*.colsouth.edu
*.colstate.edu
*.colum.edu
*.columbia.edu
*.columbiabc.edu
*.columbiaco.edu
*.columbiasc.edu
*.columbus.edu
*.com.edu
*.comillas.edu
*.commnet.edu
*.commtech.edu
*.compass.edu
*.compta.edu
*.comptiatech.edu
*.compton.edu
*.compumed.edu
*.computek.edu
*.computer.edu
*.computerman.edu
*.conception.edu
*.concord.edu
*.concorde.edu
*.concordia.edu
*.conestoga.edu
*.conncoll.edu
*.connect.edu
*.constanta.edu
*.contracosta.edu
*.contractor.edu
*.conval.edu
*.conversa.edu
*.converse.edu
*.conway.edu
*.cookman.edu
*.cooley.edu
*.cooleylaw.edu
*.cooper.edu
*.coppet.edu
*.coppin.edu
*.corban.edu
*.corcoran.edu
*.cord.edu
*.cordonbleu.edu
*.core.edu
*.cornell.edu
*.cornerstone.edu
*.corning-cc.edu
*.cornish.edu
*.corona.edu
*.cortiva.edu
*.cortland.edu
*.cos.edu
*.cosc.edu
*.cosmetology.edu
*.cosmix.edu
*.cot.edu
*.cotc.edu
*.cotf.edu
*.coto.edu
*.cottey.edu
*.courses.edu
*.covenant.edu
*.cowley.edu
*.coxcollege.edu
*.cozmo.edu
*.cpa.edu
*.cpc.edu
*.cpcc.edu
*.cpchawaii.edu
*.cpi.edu
*.cpp.edu
*.cps.edu
*.cpsphd.edu
*.cptc.edu
*.cpu.edu
*.cqlc.edu
*.cranbrook.edu
*.cranfield.edu
*.cras.edu
*.cravencc.edu
*.crbc.edu
*.crc.edu
*.crcds.edu
*.crds.edu
*.creighton.edu
*.crescent.edu
*.crestmem.edu
*.crestmont.edu
*.crhsnet.edu
*.cri.edu
*.crichton.edu
*.cril.edu
*.crimea.edu
*.crimsontech.edu
*.criswell.edu
*.crl.edu
*.crossroads.edu
*.crowder.edu
*.crown.edu
*.crtc.edu
*.crts.edu
*.cryrop.edu
*.cs.edu
*.csati.edu
*.csb.edu
*.csbradiotv.edu
*.csbs.edu
*.csbsju.edu
*.csc.edu
*.csca.edu
*.cscc.edu
*.cscu.edu
*.cse.edu
*.csf.edu
*.csftw.edu
*.cshl.edu
*.cshs.edu
*.csi.edu
*.csicollege.edu
*.csinow.edu
*.csj.edu
*.csl.edu
*.csld.edu
*.csm.edu
*.csmc.edu
*.csmd.edu
*.csn.edu
*.csny.edu
*.cso.edu
*.csp.edu
*.cspnohio.edu
*.cspp.edu
*.css.edu
*.cst.edu
*.cstcm.edu
*.cstm.edu
*.cstu.edu
*.csu.edu
*.csub.edu
*.csubak.edu
*.csuc.edu
*.csuchico.edu
*.csuci.edu
*.csudh.edu
*.csueastbay.edu
*.csufresno.edu
*.csuglobal.edu
*.csuhayward.edu
*.csulb.edu
*.csum.edu
*.csumb.edu
*.csumentor.edu
*.csun.edu
*.csuniv.edu
*.csuohio.edu
*.csupomona.edu
*.csupueblo.edu
*.csus.edu
*.csusb.edu
*.csusm.edu
*.csustan.edu
*.csusystem.edu
*.ct.edu
*.cta.edu
*.ctae.edu
*.ctc.edu
*.ctcc.edu
*.ctcd.edu
*.ctclc.edu
*.ctcpr.edu
*.ctculinary.edu
*.cte.edu
*.ctec.edu
*.cti.edu
*.ctmf.edu
*.ctres.edu
*.cts.edu
*.ctschicago.edu
*.ctsem.edu
*.ctsfw.edu
*.ctsnet.edu
*.ctstate.edu
*.ctstateu.edu
*.cttc.edu
*.ctu.edu
*.ctuonline.edu
*.ctx.edu
*.cu-portland.edu
*.cu.edu
*.cua.edu
*.cuaa.edu
*.cuanschutz.edu
*.cubt.edu
*.cuchicago.edu
*.cudenver.edu
*.cuea.edu
*.cuenet.edu
*.cuesta.edu
*.cuhk.edu
*.cui.edu
*.cuim.edu
*.cuis.edu
*.cuk.edu
*.cula.edu
*.culinary.edu
*.culver.edu
*.cumberland.edu
*.cumt.edu
*.cune.edu
*.cunef.edu
*.cunisanjuan.edu
*.cuny.edu
*.cuonline.edu
*.cup.edu
*.cure.edu
*.curry.edu
*.cursus.edu
*.curtin.edu
*.curtis.edu
*.cus.edu

*.cusat.edu
*.cusv.edu
*.cusys.edu
*.cuts.edu
*.cuw.edu
*.cuyamaca.edu
*.cv.edu
*.cva.edu
*.cvc.edu
*.cvcc.edu
*.cvccworks.edu
*.cvcs.edu
*.cvtc.edu
*.cvtech.edu
*.cvu.edu
*.cw.edu
*.cwc.edu
*.cwi.edu
*.cwpost.edu
*.cwru.edu
*.cwsl.edu
*.cwts.edu
*.cwu.edu
*.cwv.edu
*.cybertex.edu
*.cyttl.edu
*.dacc.edu
*.dadc.edu
*.dademedical.edu
*.daemen.edu
*.dalhousie.edu
*.dallas.edu
*.daltonstate.edu
*.damien-hs.edu
*.damien.edu
*.dana.edu
*.danville.edu
*.danvillecc.edu
*.darden.edu
*.dartmouth.edu
*.darton.edu
*.das.edu
*.datc.edu
*.dau.edu
*.davenport.edu
*.davidson.edu
*.davidsonccc.edu
*.davis.edu
*.davisny.edu
*.davistech.edu
*.dawson.edu
*.daybreak.edu
*.dbc.edu
*.dbcollege.edu
*.dbi.edu
*.dbq.edu
*.dbs.edu
*.dbts.edu
*.dbu.edu
*.dbumn.edu
*.dc.edu
*.dc3.edu
*.dca.edu
*.dcad.edu
*.dcb.edu
*.dcc.edu
*.dccc.edu
*.dcccd.edu
*.dcds.edu
*.dce.edu
*.dcec.edu
*.dci.edu
*.dcita.edu
*.dcs.edu
*.dct.edu
*.dctc.edu
*.dda.edu
*.ddbs.edu
*.de.edu
*.deakin.edu
*.dean.edu
*.deantech.edu
*.deanza.edu
*.debschool.edu
*.dec.edu
*.dedalo.edu
*.deepsprings.edu
*.deerfield.edu
*.defiance.edu
*.degree.edu
*.degrees.edu
*.deharttech.edu
*.dekalbtech.edu
*.delhi.edu
*.dellarte.edu
*.delmar.edu
*.delta.edu
*.deltastate.edu
*.deltatech.edu
*.delval.edu
*.deming.edu
*.denison.edu
*.denmarktech.edu
*.densem.edu
*.depaul.edu
*.depauw.edu
*.derivatives.edu
*.dermalogica.edu
*.desales.edu
*.desu.edu
*.devitt.edu
*.devry.edu
*.devrycols.edu
*.dewey.edu
*.dewv.edu
*.dgu.edu
*.dhs.edu
*.dhusson.edu
*.dia.edu
*.dibc.edu
*.dickinson.edu
*.digipen.edu
*.digital.edu
*.diku.edu
*.dillard.edu
*.dinecollege.edu
*.dinfos.edu
*.dinus.edu
*.diplomacy.edu
*.disam.edu
*.disd.edu
*.disl.edu
*.distlearn.edu
*.dit.edu
*.diu.edu
*.divinemercy.edu
*.dixie.edu
*.dixietech.edu
*.dkiapcss.edu
*.dku.edu
*.dli.edu
*.dlielc.edu
*.dliflc.edu
*.dlila.edu
*.dlu.edu
*.dmac.edu
*.dmacc.edu
*.dmartpr.edu
*.dmc.edu
*.dmce.edu
*.dmcg.edu
*.dmch.edu
*.dmgs.edu
*.dmi.edu
*.dmtc.edu
*.dmu.edu
*.dni.edu
*.doane.edu
*.dodea.edu
*.dollymonroe.edu
*.dom.edu
*.dominican.edu
*.dominion.edu
*.dominuniv.edu
*.don.edu
*.dongguk.edu
*.donnelly.edu
*.doral.edu
*.dorcas.edu
*.dordt.edu
*.dorsey.edu
*.douglasj.edu
*.dover.edu
*.dovestar.edu
*.dowling.edu
*.downstate.edu
*.doxa.edu
*.doyne.edu
*.dp.edu
*.dpc.edu
*.dpe.edu
*.dptschool.edu
*.dpu.edu
*.dragonrises.edu
*.drake.edu
*.drakestate.edu
*.dralami.edu
*.draughons.edu
*.drbu.edu
*.drew.edu
*.drexel.edu
*.drexelmed.edu
*.dri.edu
*.dru.edu
*.drury.edu
*.dsc.edu
*.dscc.edu
*.dscu.edu
*.dsdt.edu
*.dsf.edu
*.dsi.edu
*.dsiacademy.edu
*.dsl.edu
*.dslcc.edu
*.dspt.edu
*.dsu.edu
*.dtcc.edu
*.dti.edu
*.dts.edu
*.du.edu
*.ducret.edu
*.duep.edu
*.duffs.edu
*.duke.edu
*.dukeupress.edu
*.dula.edu
*.dumlao.edu
*.dunwoodie.edu
*.dunwoody.edu
*.duny.edu
*.dupage.edu
*.duq.edu
*.durhamtech.edu
*.duth.edu
*.dvc.edu
*.dvcla.edu
*.dville.edu
*.dwc-legazpi.edu
*.dwc.edu
*.dwci.edu
*.dwight.edu
*.dwu.edu
*.dxatc.edu
*.dyc.edu
*.dynamis.edu
*.dyouville.edu
*.dypatil.edu
*.ea.edu
*.eac.edu
*.eacc.edu
*.eada.edu
*.eaglecard.edu
*.eagles.edu
*.eap.edu
*.earlham.edu
*.eastcentral.edu
*.eastern.edu
*.easternct.edu
*.easternwv.edu
*.eastman.edu
*.eastms.edu
*.eastohio.edu
*.eastwest.edu
*.eastwick.edu
*.ebaf.edu
*.ebc.edu
*.ebcministry.edu
*.ebi.edu
*.ebms.edu
*.ebs.edu
*.ec.edu
*.ecam.edu
*.ecat.edu
*.ecc.edu
*.eccc.edu
*.eccu.edu
*.ecii.edu
*.eckerd.edu
*.ecla.edu
*.eclips.edu
*.ecmc.edu
*.ecok.edu
*.ecole3a.edu
*.ecoles.edu
*.ecollege.edu
*.ecology.edu
*.ecotechnics.edu
*.ecp.edu
*.ecpi.edu
*.ecpitech.edu
*.ecs.edu
*.ecsu.edu
*.ectc.edu
*.ecu.edu
*.ecua.edu
*.edcc.edu
*.edcparis.edu
*.eden.edu
*.edgeacademy.edu
*.edgecombe.edu
*.edgewood.edu
*.edhec.edu
*.ediccollege.edu
*.edinboro.edu
*.edison.edu
*.edisonohio.edu
*.edmc.edu
*.edmonds.edu
*.edpcollege.edu
*.edpschool.edu
*.eds.edu
*.edu.edu
*.education.edu
*.educatore.edu
*.educause.edu
*.educom.edu
*.educorp.edu
*.edudomains.edu
*.edukan.edu
*.edutec.edu
*.eei.edu
*.eeii.edu
*.een.edu
*.eepis-its.edu
*.ef.edu
*.efsc.edu
*.ega.edu
*.egcc.edu
*.egs.edu
*.egypt.edu
*.ehc.edu
*.ehired.edu
*.ehl.edu
*.ei.edu
*.eic.edu
*.eicc.edu
*.eicollege.edu
*.eina.edu
*.einstein.edu
*.einsteinmed.edu
*.eitan.edu
*.eitc.edu
*.eiu.edu
*.ekbdxmixkx.edu
*.eku.edu
*.elac.edu
*.elc.edu
*.elcamino.edu
*.elci.edu
*.elcok.edu
*.elearning.edu
*.elgin.edu
*.eli.edu
*.elim.edu
*.elinc.edu
*.ellis.edu
*.elmhurst.edu
*.elmira.edu
*.elms.edu
*.elon.edu
*.els.edu
*.elyon.edu
*.emba.edu
*.embryriddle.edu
*.emcc.edu
*.emedharbor.edu
*.emerson.edu
*.emetseei.edu
*.emich.edu
*.emirates.edu
*.emmanuel.edu
*.emmaus.edu
*.emory.edu
*.empcol.edu
*.emperors.edu
*.empire.edu
*.emporia.edu
*.emras.edu
*.emsom.edu
*.emu.edu
*.emuonline.edu
*.enc.edu
*.endicott.edu
*.eng4intl.edu
*.enginecity.edu
*.england.edu
*.english.edu
*.englishlci.edu
*.eni.edu
*.enmu.edu
*.ensign.edu
*.eoccc.edu
*.eoctech.edu
*.eosc.edu
*.eou.edu
*.epa.edu
*.epcc.edu
*.epei.edu
*.epfl.edu
*.epic.edu
*.epm.edu
*.erau.edu
*.erc.edu
*.erickson.edu
*.eriebc.edu
*.erieit.edu
*.eriksen.edu
*.erikson.edu
*.erskine.edu
*.erwin.edu
*.es.edu
*.esade.edu
*.esani.edu
*.esatm.edu
*.esbc.edu
*.esc-rouen.edu
*.esc.edu
*.escc.edu
*.escem.edu
*.escoffier.edu
*.ese.edu
*.esec.edu
*.eseune.edu
*.esf.edu
*.esi.edu
*.esiba.edu
*.esic.edu
*.esih.edu
*.eslacademy.edu
*.eslnyfa.edu
*.esma.edu
*.esne.edu
*.esr.edu
*.esra.edu
*.essec.edu
*.essex.edu
*.estelle.edu
*.estima.edu
*.esu.edu
*.eta.edu
*.etbu.edu
*.eteo.edu
*.etepr.edu
*.eternity.edu
*.etf.edu
*.eth.edu
*.eti.edu
*.eticampus.edu
*.eticollege.edu
*.etiopathie.edu
*.etown.edu
*.ets.edu
*.etseminary.edu
*.etsu.edu
*.ettyler.edu
*.eu.edu
*.eucon.edu
*.eunc.edu
*.eur.edu
*.eurac.edu
*.eurasia.edu
*.eurecom.edu
*.eureka.edu
*.euruni.edu
*.evangel.edu
*.evangelia.edu
*.evangelical.edu
*.evansville.edu
*.evc.edu
*.evcc.edu
*.everblue.edu
*.everest.edu
*.everett.edu
*.everettcc.edu
*.evergreen.edu
*.eversity.edu
*.evit.edu
*.evms.edu
*.ew.edu
*.ewc.edu
*.ewcollege.edu
*.ewit.edu
*.ewu.edu
*.ewubd.edu
*.ex-mba.edu
*.example.edu
*.excel.edu
*.excelsior.edu
*.exeter.edu
*.expression.edu
*.face.edu
*.facim.edu
*.facts.edu
*.facultad.edu
*.fae.edu
*.fairbanks.edu
*.fairfield.edu
*.fairhaven.edu
*.fairmount.edu
*.faith.edu
*.faithiu.edu
*.falcon.edu
*.falconihs.edu
*.famu.edu
*.fan.edu
*.fandm.edu
*.faraston.edu
*.farmingdale.edu
*.farmington.edu
*.fas.edu
*.fasttrain.edu
*.fatih.edu
*.fatihun.edu
*.fau.edu
*.faulkner.edu
*.fauser.edu
*.faytechcc.edu
*.fbcc.edu
*.fbiacademy.edu
*.fbs.edu
*.fc.edu
*.fcasd.edu
*.fcc.edu
*.fccc.edu
*.fccj.edu
*.fcim.edu
*.fcla.edu
*.fcnh.edu
*.fcps.edu
*.fcsl.edu
*.fctc.edu
*.fcts.edu
*.fcu.edu
*.fdltcc.edu
*.fdtc.edu
*.fdu.edu
*.federico.edu
*.fee.edu
*.fei.edu
*.feitian.edu
*.felician.edu
*.fergusson.edu
*.fernbank.edu
*.ferris.edu
*.ferrum.edu
*.fetc.edu
*.feu.edu
*.ffldusoe.edu
*.fgc.edu
*.fgcu.edu
*.fhchs.edu
*.fhcrc.edu
*.fhd.edu
*.fhda.edu
*.fhsu.edu
*.fhtc.edu
*.fhu.edu
*.fi.edu
*.ficu.edu
*.fidm.edu
*.fielding.edu
*.fieu.edu
*.finance.edu
*.finanzas.edu
*.findlay.edu
*.fingerlakes.edu
*.finlandia.edu
*.firn.edu
*.first.edu
*.firstschool.edu
*.fis.edu
*.fisher.edu
*.fishermore.edu
*.fisk.edu
*.fit.edu
*.fitnyc.edu
*.fitsuny.edu
*.fiu.edu
*.fivetowns.edu
*.fje.edu
*.fkcc.edu
*.fkmtc.edu
*.fknbih.edu
*.fl.edu
*.flagler.edu
*.flaglertech.edu

*.flatech.edu
*.flbc.edu
*.flbog.edu
*.flcc.edu
*.flcoll.edu
*.flet.edu
*.fletcher.edu
*.flhcon.edu
*.florida.edu
*.floridapoly.edu
*.floridatech.edu
*.flsm.edu
*.flsouthern.edu
*.fmarion.edu
*.fmc.edu
*.fmcc.edu
*.fmlh.edu
*.fms.edu
*.fmti.edu
*.fmu.edu
*.fmuniv.edu
*.fmuonline.edu
*.fnc.edu
*.fnu.edu
*.fnun.edu
*.focushope.edu
*.folger.edu
*.fomento.edu
*.fontbonne.edu
*.fontys.edu
*.foothill.edu
*.fordham.edu
*.forefront.edu
*.forest.edu
*.foresthills.edu
*.foret.edu
*.forsythtech.edu
*.fortis.edu
*.fortlewis.edu
*.fortscott.edu
*.fosbre.edu
*.fosters.edu
*.foundations.edu
*.foxcollege.edu
*.fpcc.edu
*.fpctx.edu
*.fpi.edu
*.fpp.edu
*.fps.edu
*.fptc.edu
*.fpti.edu
*.fpu.edu
*.fra.edu
*.framingham.edu
*.francis.edu
*.franciscan.edu
*.franklin.edu
*.franu.edu
*.frc.edu
*.fre3.edu
*.frederick.edu
*.fredonia.edu
*.freedom.edu
*.freenet.edu
*.freiberg.edu
*.fremont.edu
*.fresno.edu
*.fresnostate.edu
*.friends.edu
*.frontier.edu
*.frontrange.edu
*.frostburg.edu
*.frtg.edu
*.fruitland.edu
*.fsc.edu
*.fscj.edu
*.fsm.edu
*.fst.edu
*.fstm.edu
*.fsu.edu
*.fsw.edu
*.ftc.edu
*.ftcc.edu
*.ftccollege.edu
*.fti.edu
*.ftimdvadc.edu
*.ftine.edu
*.ftior.edu
*.ftium.edu
*.fts.edu
*.ftu.edu
*.fu.edu
*.fub.edu
*.fubycbtxeh.edu
*.fuesmen.edu
*.fukl.edu
*.fullcoll.edu
*.fuller.edu
*.fullerpsyc.edu
*.fullerton.edu
*.fullsail.edu
*.furman.edu
*.fus.edu
*.future.edu
*.futures.edu
*.futuretech.edu
*.fvc.edu
*.fvcc.edu
*.fvi.edu
*.fvs.edu
*.fvsu.edu
*.fvtc.edu
*.fwbbc.edu
*.fwl.edu
*.fwmep.edu
*.fxua.edu
*.ga.edu
*.gac.edu
*.gactc.edu
*.gadjahmada.edu
*.gaia.edu
*.galiano.edu
*.galileo.edu
*.gallaudet.edu
*.gannon.edu
*.garnet.edu
*.garrett.edu
*.garyounis.edu
*.gaston.edu
*.gatech.edu
*.gateway.edu
*.gatewaycc.edu
*.gatewayct.edu
*.gavilan.edu
*.gbc.edu
*.gbcnv.edu
*.gbcol.edu
*.gbi.edu
*.gbs.edu
*.gc.edu
*.gcb.edu
*.gcc.edu
*.gccaz.edu
*.gcccd.edu
*.gcccks.edu
*.gccla.edu
*.gccnj.edu
*.gcd.edu
*.gci.edu
*.gcic.edu
*.gcinter.edu
*.gcnyc.edu
*.gcs.edu
*.gcsu.edu
*.gctcok.edu
*.gctech.edu
*.gcts.edu
*.gcu.edu
*.gcuniv.edu
*.gda.edu
*.gdn.edu
*.gectcr.edu
*.geisinger.edu
*.gemini.edu
*.genealogy.edu
*.generations.edu
*.genesee.edu
*.geneseo.edu
*.genesisu.edu
*.geneva.edu
*.georgefox.edu
*.georgetown.edu
*.georgiamed.edu
*.georgian.edu
*.germanna.edu
*.getty.edu
*.gettysburg.edu
*.gfbc.edu
*.gfcmsu.edu
*.ggbts.edu
*.ggc.edu
*.ggu.edu
*.ggz.edu
*.ghc.edu
*.ghs.edu
*.ghsu.edu
*.ghusson.edu
*.gia.edu
*.gial.edu
*.gibbsboston.edu
*.gibbsnj.edu
*.gibbsny.edu
*.gibbsri.edu
*.gibbsva.edu
*.gibs.edu
*.gic.edu
*.giet.edu
*.gif.edu
*.gilman.edu
*.git.edu
*.gitam.edu
*.glasgow.edu
*.glassboro.edu
*.glbbs.edu
*.glc.edu
*.glcc.edu
*.glendale.edu
*.glendalelaw.edu
*.glenoaks.edu
*.glenville.edu
*.gli.edu
*.glion.edu
*.glit.edu
*.glitz.edu
*.global.edu
*.globaltech.edu
*.globe.edu
*.gls.edu
*.gltc.edu
*.gluck.edu
*.gm.edu
*.gma.edu
*.gmc.edu
*.gmca.edu
*.gmconline.edu
*.gmercyu.edu
*.gmh.edu
*.gmi.edu
*.gmtti.edu
*.gmu.edu
*.gnbvt.edu
*.gnomon.edu
*.gntc.edu
*.gnu.edu
*.go.edu
*.gobbc.edu
*.gocc.edu
*.gocolumbia.edu
*.gocvcc.edu
*.goddard.edu
*.gogebic.edu
*.goldengate.edu
*.goldenstate.edu
*.golfacademy.edu
*.golfcollege.edu
*.gomarquette.edu
*.gonzaga.edu
*.goodwin.edu
*.gordon.edu
*.gordonc.edu
*.gordonstate.edu
*.goshen.edu
*.gotoltc.edu
*.goucher.edu
*.gousm.edu
*.govst.edu
*.govstate.edu
*.gpc.edu
*.gps.edu
*.gptc.edu
*.gpts.edu
*.gqujtnyryl.edu
*.grace.edu
*.graceland.edu
*.graceu.edu
*.gradalis.edu
*.gram.edu
*.grandteton.edu
*.grandview.edu
*.granite.edu
*.grantcareer.edu
*.grantham.edu
*.gratz.edu
*.grayson.edu
*.grcc.edu
*.greatbay.edu
*.greatplains.edu
*.greenleaf.edu
*.greenmtn.edu
*.greenriver.edu
*.greensboro.edu
*.greenville.edu
*.greenwich.edu
*.griffintech.edu
*.griggs.edu
*.grin.edu
*.grinnell.edu
*.grooveu.edu
*.grossmont.edu
*.gru.edu
*.gs.edu
*.gsbc.edu
*.gsc.edu
*.gscollege.edu
*.gsmc.edu
*.gsot.edu
*.gspp.edu
*.gssti.edu
*.gsu.edu
*.gsw.edu
*.gt.edu
*.gtc.edu
*.gtcc.edu
*.gts.edu
*.gtu.edu
*.gtulink.edu
*.gu.edu
*.guamcc.edu
*.guardian.edu
*.guccqoekkm.edu
*.guilford.edu
*.gulfcoast.edu
*.gurnick.edu
*.gurukul.edu
*.gustavus.edu
*.gutenberg.edu
*.gvc.edu
*.gvltec.edu
*.gvsu.edu
*.gw.edu
*.gwemed.edu
*.gwinnett.edu
*.gwtech.edu
*.gwtp.edu
*.gwu.edu
*.gwumc.edu
*.ha.edu
*.ha2.edu
*.hac.edu
*.hacc.edu
*.hadley.edu
*.hahnemann.edu
*.hai.edu
*.hairpros.edu
*.halifaxcc.edu
*.hallmark.edu
*.hallym.edu
*.hamdard.edu
*.hamilton.edu
*.hamiltonia.edu
*.hamline.edu
*.hampshire.edu
*.hamptonu.edu
*.hancock.edu
*.hancocku.edu
*.handong.edu
*.haney.edu
*.hanover.edu
*.hansung.edu
*.hapjac.edu
*.harc.edu
*.harcourt.edu
*.harcum.edu
*.harding.edu
*.harford.edu
*.hargrave.edu
*.harid.edu
*.harrington.edu
*.harrisburgu.edu
*.harrison.edu
*.hartford.edu
*.hartland.edu
*.hartley.edu
*.hartnell.edu
*.hartsem.edu
*.hartwick.edu
*.harvard.edu
*.harvest.edu
*.harwich.edu
*.has.edu
*.haskell.edu
*.hastings.edu
*.hau.edu
*.hauniv.edu
*.hausson.edu
*.haven.edu
*.haverford.edu
*.hawaii.edu
*.hawaiitokai.edu
*.hawken.edu
*.hawthorn.edu
*.hawthorne.edu
*.haysacademy.edu
*.haystack.edu
*.haywood.edu
*.hazelden.edu
*.hb.edu
*.hbas.edu
*.hbc.edu
*.hbc1.edu
*.hbha.edu
*.hbmu.edu
*.hboi.edu
*.hbs.edu
*.hbu.edu
*.hbuhsd.edu
*.hc.edu
*.hcas.edu
*.hcc-nd.edu
*.hcc.edu
*.hccc.edu
*.hccfl.edu
*.hccollege.edu
*.hccpr.edu
*.hccs.edu
*.hchc.edu
*.hchs.edu
*.hci.edu
*.hcl.edu
*.hcu.edu
*.hdi.edu
*.hdmc.edu
*.hdu.edu
*.headmasters.edu
*.heald.edu
*.healdhon.edu
*.healdonline.edu
*.healthworks.edu
*.heartland.edu
*.hebisd.edu
*.hebrew.edu
*.hebron.edu
*.hec.edu
*.hecb.edu
*.heed.edu
*.heidelberg.edu
*.heights.edu
*.helenefuld.edu
*.helloworld.edu
*.helms.edu
*.hemspn.edu
*.henderson.edu
*.hendrix.edu
*.heopworks.edu
*.heriotwatt.edu
*.heritage.edu
*.heritagecs.edu
*.heritageit.edu
*.herkimer.edu
*.herzing.edu
*.hesser.edu
*.hesston.edu
*.hevs.edu
*.hfc.edu
*.hfcc.edu
*.hffheucrwr.edu
*.hfg.edu
*.hfh.edu
*.hfu.edu
*.hgc.edu
*.hgst.edu
*.hgtc.edu
*.hgu.edu
*.hh.edu
*.hhi.edu
*.hhs.edu
*.hhusson.edu
*.hi.edu
*.hibbing.edu
*.hicom.edu
*.higashi.edu
*.highland.edu
*.highlandcc.edu
*.highlands.edu
*.highline.edu
*.highpoint.edu
*.hightech.edu
*.hilbert.edu
*.hillcollege.edu
*.hillsdale.edu
*.hindscc.edu
*.hiram.edu
*.hisdon.edu
*.hisk.edu
*.hisson.edu
*.hit.edu
*.hits.edu
*.hiu.edu
*.hiwassee.edu
*.hk-ebc.edu
*.hkac.edu
*.hkapa.edu
*.hkcmi.edu
*.hksyu.edu
*.hktmc.edu
*.hku.edu
*.hlcollege.edu
*.hlg.edu
*.hm.edu
*.hmc.edu
*.hmi.edu
*.hmu.edu
*.hnu.edu
*.hocking.edu
*.hocus-lotus.edu
*.hodges.edu
*.hodson.edu
*.hofstra.edu
*.hoft.edu
*.hoganice.edu
*.hohokus.edu
*.hohokusrets.edu
*.hol.edu
*.hollins.edu
*.holmes.edu
*.holmescc.edu
*.holton-arms.edu
*.holycross.edu
*.holyfamily.edu
*.holyrosary.edu
*.homeopathy.edu
*.honam.edu
*.hondros.edu
*.hongik.edu
*.hood.edu
*.hope.edu
*.hopeonline.edu
*.hopkins-id.edu
*.hopkins.edu
*.hopkinscme.edu
*.horizon.edu
*.horizons.edu
*.horology.edu
*.hoseo.edu
*.hostos.edu
*.hotc.edu
*.hotrod.edu
*.houghton.edu
*.housatonic.edu
*.houston.edu
*.howard.edu
*.howardcc.edu
*.hpa.edu
*.hpiallied.edu
*.hptc.edu
*.hpu.edu
*.hputx.edu
*.hrcollege.edu
*.hrusson.edu
*.hs-ipabo.edu
*.hsbc.edu
*.hsc.edu
*.hscbklyn.edu
*.hscsyr.edu
*.hsi.edu
*.hson.edu
*.hss.edu
*.hsson.edu
*.hssu.edu
*.hst.edu
*.hsu.edu
*.hsutx.edu
*.htc.edu
*.hthgse.edu
*.hti.edu
*.htic.edu
*.htim.edu
*.htinj.edu
*.hts.edu
*.htu.edu
*.hu.edu
*.hu8sson.edu
*.hua.edu
*.huasan.edu
*.huason.edu
*.huc-jir.edu
*.huc.edu
*.huca.edu
*.huddon.edu
*.hudson.edu
*.hudsson.edu
*.huertas.edu
*.hufsd.edu
*.hugsr.edu
*.huhezi.edu
*.huhs.edu

*.huisson.edu
*.hult.edu
*.humak.edu
*.humboldt.edu
*.humc.edu
*.humphreys.edu
*.hun.edu
*.hunter.edu
*.huntingdon.edu
*.huntington.edu
*.huosan.edu
*.huoson.edu
*.huosson.edu
*.husn.edu
*.huson.edu
*.husoon.edu
*.husosn.edu
*.husoson.edu
*.huss.edu
*.hussianart.edu
*.hussin.edu
*.hussn.edu
*.hussno.edu
*.hussnon.edu
*.husso.edu
*.hussob.edu
*.hussom.edu
*.husson.edu
*.hussone.edu
*.hussonl.edu
*.hussonm.edu
*.hussonmedia.edu
*.hussonn.edu
*.hussons.edu
*.hussoon.edu
*.husspn.edu
*.hussson.edu
*.hussun.edu
*.hutchcc.edu
*.hutson.edu
*.huuson.edu
*.huusson.edu
*.hvcc.edu
*.hws.edu
*.hyde.edu
*.hypnosis.edu
*.iacnc.edu
*.iadc.edu
*.iadt.edu
*.iadtchicago.edu
*.iadtpitt.edu
*.iae.edu
*.iaia.edu
*.iaicu-icf.edu
*.ialf.edu
*.iam.edu
*.iama.edu
*.iamp.edu
*.iamu.edu
*.iar.edu
*.ias.edu
*.iashs.edu
*.iasi.edu
*.iastate.edu
*.iau.edu
*.iaula.edu
*.iavalley.edu
*.iaw.edu
*.iba-du.edu
*.ibais.edu
*.ibanca.edu
*.ibc.edu
*.ibc3.edu
*.ibcelpaso.edu
*.ibconline.edu
*.ibcs.edu
*.ibec.edu
*.ibero.edu
*.ibhe.edu
*.ibmc.edu
*.ibrahimieh.edu
*.ibt.edu
*.ibw.edu
*.ic.edu
*.icasi.edu
*.icb.edu
*.icbas.edu
*.icbeauty.edu
*.icc.edu
*.icchawaii.edu
*.iccic.edu
*.iccms.edu
*.iccs.edu
*.icdccollege.edu
*.ice.edu
*.iceland.edu
*.ich.edu
*.ichancellor.edu
*.iche.edu
*.ichp.edu
*.ichs.edu
*.ici.edu
*.icimss.edu
*.icls.edu
*.ico.edu
*.icoc.edu
*.icohs.edu
*.icom.edu
*.icp.edu
*.icpla.edu
*.icprjc.edu
*.icpt.edu
*.icr.edu
*.ics.edu
*.icscanada.edu
*.icseminary.edu
*.icsi.edu
*.icslearn.edu
*.icsw.edu
*.ict-ils.edu
*.ict.edu
*.ictc.edu
*.ictctech.edu
*.ictech.edu
*.icu.edu
*.idaho.edu
*.idb.edu
*.idbsu.edu
*.idc.edu
*.idec.edu
*.idi.edu
*.idm.edu
*.idsva.edu
*.idt.edu
*.idti.edu
*.idu.edu
*.idv.edu
*.ie.edu
*.iec-dvc.edu
*.iec-occ.edu
*.iecc.edu
*.iecs.edu
*.ied.edu
*.iede.edu
*.iei.edu
*.ies.edu
*.iesa.edu
*.iese.edu
*.ieside.edu
*.ietlucknow.edu
*.iewu.edu
*.ifam.edu
*.ifti.edu
*.iga.edu
*.iglobal.edu
*.ignatius.edu
*.igu.edu
*.igw.edu
*.ihcp.edu
*.ihe.edu
*.ihmctan.edu
*.ihp.edu
*.iht.edu
*.ii.edu
*.iias.edu
*.iic.edu
*.iicm.edu
*.iicusa.edu
*.iif.edu
*.iifa.edu
*.iifbs.edu
*.iift.edu
*.iihcp.edu
*.iile.edu
*.iilm.edu
*.iip.edu
*.iipm.edu
*.iirp.edu
*.iiswbm.edu
*.iit.edu
*.iitb.edu
*.iitnj.edu
*.iitr.edu
*.ijtrmdhwuo.edu
*.ila.edu
*.ilc.edu
*.ili.edu
*.iliff.edu
*.ilisagvik.edu
*.illinois.edu
*.illinpos.edu
*.ilm.edu
*.ilsc.edu
*.ilslaw.edu
*.ilstu.edu
*.ilt.edu
*.ilternet.edu
*.ilu.edu
*.ilws.edu
*.imbc.edu
*.imc.edu
*.imdr.edu
*.ime.edu
*.imed.edu
*.imf.edu
*.imgs-coh.edu
*.imi.edu
*.imm.edu
*.immaculata.edu
*.impachawaii.edu
*.impacu.edu
*.imperial.edu
*.imr.edu
*.ims.edu
*.imsa.edu
*.imt.edu
*.imti.edu
*.imtu.edu
*.imu.edu
*.incae.edu
*.india.edu
*.indiana.edu
*.indianatech.edu
*.indianhills.edu
*.indstate.edu
*.indwes.edu
*.indycc.edu
*.infoteam.edu
*.infotech.edu
*.ingram.edu
*.inha.edu
*.inhe.edu
*.inje.edu
*.inlingua-if.edu
*.inmantec.edu
*.inovatech.edu
*.insead.edu
*.insession.edu
*.inste.edu
*.int.edu
*.intania.edu
*.intec.edu
*.integrity.edu
*.intellitec.edu
*.inter.edu
*.interboro.edu
*.intercoast.edu
*.intercol.edu
*.interface.edu
*.interlink.edu
*.intermetro.edu
*.internachi.edu
*.internet2.edu
*.internexus.edu
*.interponce.edu
*.intersg.edu
*.interstudi.edu
*.intl.edu
*.intrax.edu
*.inverhills.edu
*.invivo.edu
*.iohs.edu
*.iom.edu
*.iona.edu
*.ionacollege.edu
*.iot.edu
*.iotm.edu
*.iou.edu
*.iowacentral.edu
*.iowalakes.edu
*.iowaregents.edu
*.ipag.edu
*.ipfw.edu
*.ipm.edu
*.ipr.edu
*.ips.edu
*.ipsb.edu
*.ipsciences.edu
*.ipu.edu
*.iqe.edu
*.iqs.edu
*.iradio.edu
*.irenes.edu
*.iris.edu
*.irsc.edu
*.iru.edu
*.irus.edu
*.irvine.edu
*.is.edu
*.isb.edu
*.isce.edu
*.iscm.edu
*.isec.edu
*.ishb.edu
*.ishbt.edu
*.ishc.edu
*.isi.edu
*.islam.edu
*.ism.edu
*.ismett.edu
*.iso.edu
*.isothermal.edu
*.isparis.edu
*.iss.edu
*.issa.edu
*.issaco.edu
*.issaonline.edu
*.issnschool.edu
*.issweb.edu
*.ist.edu
*.istc.edu
*.ists.edu
*.istu.edu
*.isu.edu
*.isunet.edu
*.it-colleges.edu
*.it.edu
*.ita.edu
*.itap.edu
*.itascacc.edu
*.itb.edu
*.itc.edu
*.itcmiami.edu
*.itcpr.edu
*.itdc.edu
*.itdt.edu
*.itea.edu
*.itec.edu
*.itech.edu
*.itesm.edu
*.itf.edu
*.ithaca.edu
*.iti.edu
*.iticollege.edu
*.itm.edu
*.itmindia.edu
*.itp.edu
*.itpasia.edu
*.its-sby.edu
*.its.edu
*.itsla.edu
*.itsptrliup.edu
*.itt-tech.edu
*.itt.edu
*.itttech.edu
*.itu.edu
*.iu.edu
*.iuav.edu
*.iub.edu
*.iubat.edu
*.iucis.edu
*.iue.edu
*.iufs.edu
*.iufw.edu
*.iugaza.edu
*.iuhs.edu
*.iuk.edu
*.iukenya.edu
*.ium.edu
*.iun.edu
*.iuonline.edu
*.iup.edu
*.iupmsdiup.edu
*.iups.edu
*.iupuc.edu
*.iupui.edu
*.ius.edu
*.iusb.edu
*.iut-dhaka.edu
*.iutepi.edu
*.iuvienna.edu
*.ivc.edu
*.ivcc.edu
*.ivs.edu
*.ivy.edu
*.ivytech.edu
*.iw.edu
*.iwc.edu
*.iwcc.edu
*.iwp.edu
*.iws.edu
*.iwu.edu
*.iyrs.edu
*.jadavpur.edu
*.jalc.edu
*.jamessprunt.edu
*.jarvis.edu
*.javelin.edu
*.javier.edu
*.jazz.edu
*.jba.edu
*.jbc.edu
*.jbims.edu
*.jbnu.edu
*.jbs.edu
*.jbu.edu
*.jc.edu
*.jcas.edu
*.jcc.edu
*.jccc.edu
*.jccmi.edu
*.jchs.edu
*.jcjc.edu
*.jcm.edu
*.jcollege.edu
*.jcsu.edu
*.jcu.edu
*.jdc.edu
*.jdcc.edu
*.jec.edu
*.jed.edu
*.jeffco.edu
*.jefferson.edu
*.jem.edu
*.jessup.edu
*.jetpower.edu
*.jets.edu
*.jett.edu
*.jeunes.edu
*.jewell.edu
*.jfku.edu
*.jh.edu
*.jhbc.edu
*.jhmi.edu
*.jhsph.edu
*.jhu.edu
*.jhuapl.edu
*.jinnah.edu
*.jipmer.edu
*.jiu.edu
*.jiwaji.edu
*.jjc.edu
*.jku.edu
*.jmc.edu
*.jmls.edu
*.jmu.edu
*.jmvu.edu
*.jn.edu
*.jna.edu
*.jnmc.edu
*.johncabot.edu
*.johnjay.edu
*.johnpaolo.edu
*.johnpaulii.edu
*.johnson.edu
*.johnsonu.edu
*.johnstoncc.edu
*.johnwesley.edu
*.jones.edu
*.joyce.edu
*.jpcatholic.edu
*.jpu.edu
*.jrenee.edu
*.jru.edu
*.jsbc.edu
*.jsc.edu
*.jscc.edu
*.jsou.edu
*.jstb.edu
*.jsu.edu
*.jsums.edu
*.jtcc.edu
*.jts.edu
*.jtsa.edu
*.ju.edu
*.juc.edu
*.judson-il.edu
*.judson.edu
*.judsonu.edu
*.juilliard.edu
*.julliard.edu
*.julphar.edu
*.jung.edu
*.jungtao.edu
*.juniata.edu
*.juniv.edu
*.jvbrown.edu
*.jwc.edu
*.jwcc.edu
*.jwu.edu
*.k-state.edu
*.kairos.edu
*.kaist.edu
*.kanisius.edu
*.kansas.edu
*.kansascity.edu
*.kaplan.edu
*.kaplanu.edu
*.karishma.edu
*.karunya.edu
*.kaskaskia.edu
*.kau.edu
*.kba.edu
*.kbc.edu
*.kbocc.edu
*.kc.edu
*.kcad.edu
*.kcai.edu
*.kcc.edu
*.kccbs.edu
*.kccd.edu
*.kccm.edu
*.kcg.edu
*.kchair.edu
*.kciti.edu
*.kckcc.edu
*.kcma.edu
*.kctcs.edu
*.kcu.edu
*.kcumb.edu
*.kdstudio.edu
*.kdu.edu
*.kean.edu
*.kedge.edu
*.kee.edu
*.keene.edu
*.keio.edu
*.keiseru.edu
*.keller.edu
*.kellogg.edu
*.kem.edu
*.kendall.edu
*.kennesaw.edu
*.kenrick.edu
*.kent-school.edu
*.kent.edu
*.kentlaw.edu
*.kentucky.edu
*.kenyon.edu
*.kepler.edu
*.kernel.edu
*.kestrel.edu
*.ketchum.edu
*.kettering.edu
*.keuka.edu
*.keycollege.edu
*.keystone.edu
*.kgi.edu
*.kgmcindia.edu
*.khai.edu
*.kharkiv.edu
*.khs.edu
*.khu.edu
*.kianvirtual.edu
*.kids.edu
*.kiet.edu
*.kilgore.edu
*.kilian.edu
*.kimberly.edu
*.king.edu
*.kings.edu
*.kingston.edu
*.kingstonu.edu
*.kingsway.edu
*.kingswood.edu
*.kinocollege.edu
*.kirameki.edu
*.kirkwood.edu
*.kiropraktik.edu
*.kirtland.edu
*.kish.edu
*.kit.edu
*.kits.edu
*.kku.edu
*.klamathcc.edu
*.klcasjqnfj.edu
*.klepharm.edu
*.klnce.edu
*.klove.edu
*.klsimer.edu
*.kmbc.edu
*.kmcemed.edu
*.kmsd.edu
*.knight.edu
*.knmiet.edu
*.knox.edu

*.knupe.edu
*.kona.edu
*.kongu.edu
*.korea.edu
*.kosin.edu
*.kpsahs.edu
*.krasnoyarsk.edu
*.ksbe.edu
*.ksds.edu
*.ksh.edu
*.ksi.edu
*.kspu.edu
*.ksrcas.edu
*.kstone.edu
*.ksu.edu
*.ktc.edu
*.kti.edu
*.ktu.edu
*.ku.edu
*.kucampus.edu
*.kumc.edu
*.kuniv.edu
*.kusd.edu
*.kutztown.edu
*.kuwait.edu
*.kuyper.edu
*.kvcc.edu
*.kvctc.edu
*.kwantlen.edu
*.kwc.edu
*.kwu.edu
*.kysu.edu
*.kytc.edu
*.kyunghee.edu
*.kzoo.edu
*.kzuocyiluz.edu
*.la.edu
*.laafa.edu
*.laba.edu
*.labette.edu
*.labfour.edu
*.labi.edu
*.labiomed.edu
*.laboure.edu
*.lac.edu
*.laca.edu
*.lacademie.edu
*.lacademy.edu
*.lacasita.edu
*.lacc.edu
*.laccd.edu
*.lackawanna.edu
*.lacm.edu
*.lacoe.edu
*.lacollege.edu
*.lacs.edu
*.ladelta.edu
*.lado.edu
*.lafayette.edu
*.lafilm.edu
*.lagrange.edu
*.laguardia.edu
*.lahc.edu
*.lajames.edu
*.lakecitycc.edu
*.lakeerie.edu
*.lakeforest.edu
*.lakeland.edu
*.lakelandcc.edu
*.lakeside.edu
*.lakeviewcol.edu
*.lakewood.edu
*.lama.edu
*.lamar.edu
*.lamarcc.edu
*.lamarpa.edu
*.lambert.edu
*.lambuth.edu
*.lamission.edu
*.lamson.edu
*.lander.edu
*.landmark.edu
*.landuselaw.edu
*.lane.edu
*.lanecc.edu
*.lanecollege.edu
*.laney.edu
*.lang.edu
*.langston.edu
*.laniertech.edu
*.lansbridge.edu
*.laort.edu
*.lapacific.edu
*.lapietra.edu
*.lapu.edu
*.laredo.edu
*.laregents.edu
*.lareine.edu
*.larenstein.edu
*.laroche.edu
*.lasalle.edu
*.lasalletech.edu
*.lasc.edu
*.lasdallas.edu
*.lasell.edu
*.lasierra.edu
*.latech.edu
*.latrobe.edu
*.lattc.edu
*.lau.edu
*.laurel.edu
*.laurelridge.edu
*.laurus.edu
*.lausanne.edu
*.lavc.edu
*.laverne.edu
*.law.edu
*.lawrence.edu
*.lawsonstate.edu
*.lbc.edu
*.lbcc.edu
*.lbcihouma.edu
*.lbhc.edu
*.lbi.edu
*.lbs.edu
*.lbsu.edu
*.lbts.edu
*.lbu.edu
*.lbwcc.edu
*.lc.edu
*.lca.edu
*.lcad.edu
*.lcc.edu
*.lccc.edu
*.lcchs.edu
*.lcctc.edu
*.lci.edu
*.lclark.edu
*.lcmc.edu
*.lcn.edu
*.lco.edu
*.lcsc.edu
*.lctcs.edu
*.lctcsonline.edu
*.lcu.edu
*.lcus.edu
*.ldsbc.edu
*.ldsces.edu
*.leaders.edu
*.leadership.edu
*.learey.edu
*.learn.edu
*.learnet.edu
*.learnquest.edu
*.lec.edu
*.lecole.edu
*.lecom.edu
*.lee.edu
*.leecollege.edu
*.leeds.edu
*.leeu.edu
*.lehigh.edu
*.lehman.edu
*.leiden.edu
*.leland.edu
*.lemoyne.edu
*.lenoircc.edu
*.les.edu
*.lesley.edu
*.lesroches.edu
*.letourneau.edu
*.letran.edu
*.letu.edu
*.lewis.edu
*.lewisu.edu
*.lextheo.edu
*.lfc.edu
*.lfcc.edu
*.lfgsm.edu
*.lhc.edu
*.lhup.edu
*.lhupclfd.edu
*.liba.edu
*.liberty.edu
*.libertyjr.edu
*.libi.edu
*.librty.edu
*.libs.edu
*.liceo.edu
*.liceomaffei.edu
*.licm.edu
*.life-east.edu
*.life.edu
*.lifelinefl.edu
*.lifepacific.edu
*.lifewest.edu
*.lifework.edu
*.lighthouse.edu
*.lij.edu
*.limcollege.edu
*.limestone.edu
*.linc.edu
*.lincoln.edu
*.lincolninst.edu
*.lincolnlaw.edu
*.lincolntech.edu
*.lincolnu.edu
*.lincolnuca.edu
*.lindenwood.edu
*.lindsey.edu
*.lineman.edu
*.linfield.edu
*.lingua.edu
*.linnbenton.edu
*.linnstate.edu
*.lintonhall.edu
*.linux.edu
*.lion.edu
*.lionel.edu
*.lipscomb.edu
*.lisma.edu
*.lit.edu
*.litexas.edu
*.litr.edu
*.littlehoop.edu
*.liu.edu
*.liunet.edu
*.livelytech.edu
*.livingstone.edu
*.ljcrf.edu
*.ljic.edu
*.llcc.edu
*.llim.edu
*.lls.edu
*.lltc.edu
*.llu.edu
*.llumc.edu
*.lmc.edu
*.lmh.edu
*.lmi.edu
*.lmu.edu
*.lmunet.edu
*.loc.edu
*.lockhaven.edu
*.logan.edu
*.loganlib.edu
*.logos.edu
*.london.edu
*.lonestar.edu
*.longwood.edu
*.longy.edu
*.lonmorris.edu
*.loopback.edu
*.lorainccc.edu
*.loras.edu
*.lorma.edu
*.losmedanos.edu
*.losrios.edu
*.louisburg.edu
*.louisiana.edu
*.louisville.edu
*.lourdes.edu
*.louvre.edu
*.love.edu
*.lowell.edu
*.loyno.edu
*.loyola.edu
*.loyolahs.edu
*.loyolanet.edu
*.lpts.edu
*.lr.edu
*.lrcc.edu
*.lrcs.edu
*.lrsc.edu
*.lru.edu
*.lsb.edu
*.lsc.edu
*.lscc.edu
*.lsco.edu
*.lse.edu
*.lsi.edu
*.lsmsa.edu
*.lspr.edu
*.lsps.edu
*.lssc.edu
*.lssu.edu
*.lst.edu
*.lstc.edu
*.lsu.edu
*.lsua.edu
*.lsue.edu
*.lsuhealth.edu
*.lsuhs.edu
*.lsuhsc-s.edu
*.lsuhsc.edu
*.lsuhscs.edu
*.lsumc.edu
*.lsus.edu
*.lsusystem.edu
*.ltc.edu
*.ltcc.edu
*.lternet.edu
*.lti.edu
*.ltionline.edu
*.ltlqfqufvj.edu
*.lts.edu
*.ltsg.edu
*.ltsp.edu
*.ltss.edu
*.ltu.edu
*.lubavitch.edu
*.luc.edu
*.luiss.edu
*.lumbeeriver.edu
*.lumc.edu
*.lumcon.edu
*.lumos.edu
*.luna.edu
*.lunet.edu
*.luofc.edu
*.luresext.edu
*.luther.edu
*.lutherrice.edu
*.luthersem.edu
*.luzerne.edu
*.lv.edu
*.lvc.edu
*.lvcollege.edu
*.lwit.edu
*.lws.edu
*.lwtc.edu
*.lwtech.edu
*.lycoming.edu
*.lymeacademy.edu
*.lynchburg.edu
*.lyndonstate.edu
*.lynn.edu
*.lyon.edu
*.lytlesrebc.edu
*.ma.edu
*.mabts.edu
*.mabtsne.edu
*.mac.edu
*.macalester.edu
*.macalstr.edu
*.macc.edu
*.maccormac.edu
*.macedonia.edu
*.machias.edu
*.mackenzie.edu
*.macohs.edu
*.macomb.edu
*.maconstate.edu
*.macquarie.edu
*.macu.edu
*.madonna.edu
*.maejo.edu
*.magdalen.edu
*.magee.edu
*.magellan.edu
*.magnet.edu
*.magnolia.edu
*.maharishi.edu
*.mahidol.edu
*.mai.edu
*.maia.edu
*.maimonides.edu
*.maine.edu
*.mainemedia.edu
*.malone.edu
*.mamak.edu
*.manateetech.edu
*.manc.edu
*.manchester.edu
*.mancol.edu
*.mancosre6.edu
*.mandl.edu
*.manhattan.edu
*.manipal.edu
*.manna.edu
*.mannes.edu
*.manor.edu
*.mansfield.edu
*.mantech.edu
*.manthano.edu
*.manu.edu
*.maranatha.edu
*.mariacy.edu
*.marian.edu
*.marianas.edu
*.mariancourt.edu
*.maricopa.edu
*.marietta.edu
*.marin.edu
*.marinello.edu
*.mariontc.edu
*.marist.edu
*.maritime.edu
*.marlboro.edu
*.marquette.edu
*.marshall.edu
*.marshill.edu
*.martin.edu
*.martincc.edu
*.martinus.edu
*.marybaldwin.edu
*.marygrove.edu
*.maryland.edu
*.marylhurst.edu
*.marymount.edu
*.marymountpv.edu
*.maryville.edu
*.marywood.edu
*.mash.edu
*.mass.edu
*.massart.edu
*.massasoit.edu
*.massbay.edu
*.masscomm.edu
*.massey.edu
*.massmentor.edu
*.masters.edu
*.matarazzo.edu
*.matc.edu
*.matcmadison.edu
*.matech.edu
*.math.edu
*.mau.edu
*.maufl.edu
*.mayanot.edu
*.mayim.edu
*.mayland.edu
*.mayo.edu
*.mba.edu
*.mbbc.edu
*.mbc.edu
*.mbcs.edu
*.mbhs.edu
*.mbi.edu
*.mbiyeshiva.edu
*.mbl.edu
*.mbs.edu
*.mbseminary.edu
*.mbtipr.edu
*.mbts.edu
*.mbu.edu
*.mc.edu
*.mc3.edu
*.mca.edu
*.mcad.edu
*.mcb-seattle.edu
*.mcbc.edu
*.mcbi.edu
*.mcc.edu
*.mccann.edu
*.mccb.edu
*.mccc.edu
*.mccd.edu
*.mcckc.edu
*.mccks.edu
*.mccloskey.edu
*.mccn.edu
*.mccneb.edu
*.mccnh.edu
*.mccollege.edu
*.mccormick.edu
*.mccsc.edu
*.mcdaniel.edu
*.mcdonough.edu
*.mced.edu
*.mcg.edu
*.mcgeorge.edu
*.mcgill.edu
*.mcgregor.edu
*.mchenry.edu
*.mchp.edu
*.mchs.edu
*.mci.edu
*.mcidenver.edu
*.mcinj.edu
*.mciot.edu
*.mcjvs.edu
*.mckendree.edu
*.mckenzie.edu
*.mcla.edu
*.mclennan.edu
*.mcm.edu
*.mcn.edu
*.mcneese.edu
*.mcny.edu
*.mcon.edu
*.mcp.edu
*.mcpherson.edu
*.mcphs.edu
*.mcphu.edu
*.mcs.edu
*.mctc.edu
*.mcti.edu
*.mcts.edu
*.mcu.edu
*.mcvh-vcu.edu
*.mcvs.edu
*.mcw.edu
*.md.edu
*.mdanderson.edu
*.mdc.edu
*.mdi.edu
*.mdivs.edu
*.me.edu
*.meadville.edu
*.mec.edu
*.meca.edu
*.mecc.edu
*.mechtech.edu
*.mecollege.edu
*.mecr.edu
*.med.edu
*.medacademy.edu
*.medaille.edu
*.medcc.edu
*.medcentral.edu
*.mediatech.edu
*.medical.edu
*.medicina.edu
*.medixschool.edu
*.medtech.edu
*.meduohio.edu
*.medvance.edu
*.mei.edu
*.melbourne.edu
*.melrose.edu
*.memphis.edu
*.mendo.edu
*.mendocino.edu
*.menlo.edu
*.menninger.edu
*.menominee.edu
*.mephi.edu
*.mercer.edu
*.mercersburg.edu
*.merchia.edu
*.mercury.edu
*.mercy.edu
*.mercyhurst.edu
*.mercynet.edu
*.meredith.edu
*.meridian.edu
*.meridiancc.edu
*.meridianu.edu
*.merit.edu
*.meritu.edu
*.merkaz.edu
*.merrimack.edu
*.merritt.edu
*.merryfield.edu
*.meru.edu
*.mesa.edu
*.mesabirange.edu
*.mesacc.edu
*.mesalands.edu
*.mesastate.edu
*.mesls.edu
*.messiah.edu
*.met.edu
*.metacenter.edu
*.metc.edu
*.meteo.edu
*.methodist.edu
*.metnet.edu
*.metro.edu
*.metrostate.edu
*.metrotech.edu
*.metu.edu
*.meu.edu
*.mfc.edu
*.mfhs.edu

*.mfldclin.edu
*.mfti.edu
*.mfwi.edu
*.mga.edu
*.mgc.edu
*.mgcc.edu
*.mgccc.edu
*.mghihp.edu
*.mgs.edu
*.mgu.edu
*.mgupi.edu
*.mgv.edu
*.mhc.edu
*.mhcc.edu
*.mhg.edu
*.mhgs.edu
*.mhpcc.edu
*.mhrc.edu
*.mhu.edu
*.mi.edu
*.miad.edu
*.miami.edu
*.miamijacobs.edu
*.miamilakes.edu
*.miamioh.edu
*.mias.edu
*.miat.edu
*.mib.edu
*.mica.edu
*.michlala.edu
*.michlalah.edu
*.micpel.edu
*.micro.edu
*.mid-america.edu
*.midamerica.edu
*.middlebury.edu
*.middlesex.edu
*.middlesexcc.edu
*.midfieldic.edu
*.midland.edu
*.midlandu.edu
*.midmich.edu
*.midpac.edu
*.midsouthcc.edu
*.midstate.edu
*.midway.edu
*.midwest.edu
*.midwestern.edu
*.midwesttech.edu
*.midwifery.edu
*.miet.edu
*.miis.edu
*.miles.edu
*.milescc.edu
*.millennium.edu
*.millermotte.edu
*.millersv.edu
*.milligan.edu
*.millikin.edu
*.mills.edu
*.millsaps.edu
*.milton.edu
*.mim.edu
*.mimh.edu
*.mimusa.edu
*.mindbody.edu
*.mindless.edu
*.mineralarea.edu
*.minerva.edu
*.mines.edu
*.minneapolis.edu
*.minnesota.edu
*.minnstate.edu
*.minotstateu.edu
*.mints.edu
*.miracosta.edu
*.mise.edu
*.miskatonic.edu
*.mispp.edu
*.missio.edu
*.mississippi.edu
*.missouri.edu
*.mit.edu
*.mita.edu
*.mitchell.edu
*.mitchellcc.edu
*.mitchells.edu
*.mitindia.edu
*.mitpr.edu
*.mits.edu
*.miu.edu
*.mizpa.edu
*.mizzou.edu
*.mjc.edu
*.mji.edu
*.mkecc.edu
*.mkgacademy.edu
*.mlatc.edu
*.mlaw.edu
*.mlb.edu
*.mlc-wels.edu
*.mlc.edu
*.mli.edu
*.mliesl.edu
*.mlliberty.edu
*.mm.edu
*.mma.edu
*.mmc.edu
*.mmci.edu
*.mmi.edu
*.mmitech.edu
*.mmm.edu
*.mmpv.edu
*.mmri.edu
*.mmrl.edu
*.mmui.edu
*.mnscu.edu
*.mnsfld.edu
*.mnsmc.edu
*.mnstate.edu
*.mnsu.edu
*.mntc.edu
*.mnu.edu
*.mnwest.edu
*.mobap.edu
*.moc.edu
*.mocn.edu
*.modern.edu
*.mohave.edu
*.molloy.edu
*.molview.edu
*.monaco.edu
*.monash.edu
*.mondejar.edu
*.mondragon.edu
*.monm.edu
*.monmouth.edu
*.monomoy.edu
*.monroe-cc.edu
*.monroe.edu
*.monroe2cwd.edu
*.monroecc.edu
*.monroeccc.edu
*.monroecoll.edu
*.montana.edu
*.montanacc.edu
*.montcalm.edu
*.montclair.edu
*.monteclaro.edu
*.monterey.edu
*.montereybay.edu
*.montereylaw.edu
*.montessori.edu
*.montevallo.edu
*.montgomery.edu
*.montreat.edu
*.montserrat.edu
*.moody.edu
*.moore.edu
*.mooretech.edu
*.morainepark.edu
*.moravian.edu
*.morehead-st.edu
*.morehead.edu
*.morehouse.edu
*.moreland.edu
*.morgan.edu
*.morgancc.edu
*.morningside.edu
*.morris.edu
*.morrisbrown.edu
*.morrisville.edu
*.morthland.edu
*.morton.edu
*.mosis.edu
*.motech.edu
*.motherindia.edu
*.motlow.edu
*.motqahzduw.edu
*.mount.edu
*.mountcarmel.edu
*.mountida.edu
*.mountmarty.edu
*.mountolive.edu
*.mountunion.edu
*.moval.edu
*.mpc.edu
*.mpcc.edu
*.mpgu.edu
*.mpi.edu
*.mpm.edu
*.mpow.edu
*.mrc.edu
*.mrilearning.edu
*.mru.edu
*.ms.edu
*.msa.edu
*.msae.edu
*.msb.edu
*.msbbcs.edu
*.msbcollege.edu
*.msc.edu
*.mscc.edu
*.msccollege.edu
*.mscd.edu
*.mscjc.edu
*.mscok.edu
*.msd.edu
*.msdelta.edu
*.msj.edu
*.msjc.edu
*.msjnet.edu
*.msl.edu
*.mslaw.edu
*.msm.edu
*.msmary.edu
*.msmc.edu
*.msmnyc.edu
*.msmu.edu
*.msoe.edu
*.msp.edu
*.mspp.edu
*.msqa.edu
*.msrchm.edu
*.msrit.edu
*.mssm.edu
*.msstate.edu
*.mssu.edu
*.mst.edu
*.mstc.edu
*.mstsc.edu
*.msu-b.edu
*.msu.edu
*.msubillings.edu
*.msudenver.edu
*.msugf.edu
*.msun.edu
*.msus.edu
*.msutexas.edu
*.mtaloy.edu
*.mtangel.edu
*.mtc.edu
*.mtec.edu
*.mtech.edu
*.mtecwv.edu
*.mtholyoke.edu
*.mti-jatt.edu
*.mti.edu
*.mtic.edu
*.mticollege.edu
*.mtiweb.edu
*.mtj.edu
*.mtmary.edu
*.mtmc.edu
*.mtmercy.edu
*.mts.edu
*.mtsa.edu
*.mtsac.edu
*.mtsamerica.edu
*.mtscampus.edu
*.mtschool.edu
*.mtsierra.edu
*.mtso.edu
*.mtstmary.edu
*.mtsu.edu
*.mtti.edu
*.mtu.edu
*.mtwilson.edu
*.mu.edu
*.mua.edu
*.muc.edu
*.mud.edu
*.mueller.edu
*.muhlenberg.edu
*.muhs.edu
*.muih.edu
*.mukogawa.edu
*.multitrex.edu
*.multnomah.edu
*.mum.edu
*.munet.edu
*.muohio.edu
*.murraystate.edu
*.mus.edu
*.musc.edu
*.muskegoncc.edu
*.muskingum.edu
*.must.edu
*.muw.edu
*.mvc.edu
*.mvcc.edu
*.mville.edu
*.mvnu.edu
*.mvsu.edu
*.mvu.edu
*.mw.edu
*.mwai.edu
*.mwcc.edu
*.mwcollege.edu
*.mwmcarey.edu
*.mwpai.edu
*.mwpi.edu
*.mwsu.edu
*.mwzgjuqtfz.edu
*.mxcc.edu
*.mxschool.edu
*.mybrcc.edu
*.mycc.edu
*.mycci.edu
*.mycenae.edu
*.myconcorde.edu
*.myfiaa.edu
*.myhusson.edu
*.myita.edu
*.myiupmyiup.edu
*.mylbc.edu
*.mylincoln.edu
*.myltc.edu
*.mymiu.edu
*.myneltc.edu
*.myotherapy.edu
*.myptc.edu
*.mysmu.edu
*.myunion.edu
*.myutpb.edu
*.myy.edu
*.na.edu
*.naa.edu
*.naal.edu
*.naba.edu
*.nabc.edu
*.nac.edu
*.nacc.edu
*.nacka.edu
*.nacu.edu
*.nadc.edu
*.nae.edu
*.naes.edu
*.naic.edu
*.naicu.edu
*.nair.edu
*.najah.edu
*.nam.edu
*.nap.edu
*.napavalley.edu
*.napmed.edu
*.naps.edu
*.naropa.edu
*.nas.edu
*.nash.edu
*.nashcc.edu
*.nashotah.edu
*.nashua.edu
*.nashuacc.edu
*.nasm.edu
*.nassonu.edu
*.nasx.edu
*.national.edu
*.nationalpti.edu
*.nationaltax.edu
*.nationsu.edu
*.natlcollege.edu
*.natpoly.edu
*.natuniv.edu
*.nau.edu
*.nauonline.edu
*.navajotech.edu
*.navarra.edu
*.navy.edu
*.nayanova.edu
*.naz.edu
*.nazarene.edu
*.nazareth.edu
*.nba.edu
*.nbc.edu
*.nbi.edu
*.nbs.edu
*.nbss.edu
*.nbtc.edu
*.nbti.edu
*.nbts.edu
*.nbusd.edu
*.nc.edu
*.nca.edu
*.ncad.edu
*.ncat.edu
*.ncbc.edu
*.ncbt.edu
*.ncc.edu
*.nccc.edu
*.ncce.edu
*.nccs.edu
*.nccu.edu
*.nccusa.edu
*.nce.edu
*.ncf.edu
*.ncfm.edu
*.nche.edu
*.nci.edu
*.ncifti.edu
*.ncioh.edu
*.ncit.edu
*.ncktc.edu
*.ncmc.edu
*.ncmich.edu
*.ncmissouri.edu
*.ncnm.edu
*.ncpe.edu
*.ncsa.edu
*.ncsc.edu
*.ncseaa.edu
*.ncssm.edu
*.ncstate.edu
*.ncstrades.edu
*.ncsu.edu
*.nctc.edu
*.ncti.edu
*.ncu.edu
*.ncuindia.edu
*.ncura.edu
*.ncwc.edu
*.nd.edu
*.ndc.edu
*.ndic.edu
*.ndm.edu
*.ndmu.edu
*.ndnu.edu
*.nds.edu
*.ndscs.edu
*.ndsigis.edu
*.ndsion.edu
*.ndsu.edu
*.ndu.edu
*.ndus.edu
*.ne.edu
*.near.edu
*.nebc.edu
*.nebo.edu
*.nebraska.edu
*.nec.edu
*.necb.edu
*.necc.edu
*.nechristian.edu
*.neci.edu
*.necmusic.edu
*.neco.edu
*.necsi.edu
*.negst.edu
*.nehc.edu
*.neit.edu
*.neiu.edu
*.nel.edu
*.nemcc.edu
*.neo.edu
*.neomed.edu
*.neosho.edu
*.neoucom.edu
*.nes.edu
*.nesa.edu
*.nescom.edu
*.nese.edu
*.neshd.edu
*.nesl.edu
*.nesop.edu
*.netc.edu
*.netech.edu
*.netschool.edu
*.nettts.edu
*.neu.edu
*.neumann.edu
*.neumont.edu
*.nevada.edu
*.new-zealand.edu
*.new.edu
*.newarka.edu
*.newberry.edu
*.newbold.edu
*.newbury.edu
*.newcollege.edu
*.newcovenant.edu
*.newgate.edu
*.newhaven.edu
*.newhope.edu
*.newhorizons.edu
*.newlife.edu
*.newman.edu
*.newmanu.edu
*.newpaltz.edu
*.newport.edu
*.newriver.edu
*.newschool.edu
*.newton.edu
*.newzealand.edu
*.nfc.edu
*.nfcc.edu
*.nfi.edu
*.nfo.edu
*.nftc.edu
*.ngcsu.edu
*.ngi.edu
*.ngihca.edu
*.ngs.edu
*.ngu.edu
*.nhc.edu
*.nhcc.edu
*.nhed.edu
*.nhi.edu
*.nhia.edu
*.nhnc.edu
*.nhsc.edu
*.nhti.edu
*.nhu.edu
*.ni-u.edu
*.ni.edu
*.niacc.edu
*.niagara.edu
*.nic.edu
*.nicc.edu
*.nich.edu
*.nicholls.edu
*.nichols.edu
*.nid.edu
*.nightingale.edu
*.nila.edu
*.nild.edu
*.nim.edu
*.nima.edu
*.nimaa.edu
*.nipissing.edu
*.nirc.edu
*.nist.edu
*.nitie.edu
*.nitt.edu
*.niu.edu
*.njc.edu
*.njcu.edu
*.njit.edu
*.nku.edu
*.nl.edu
*.nlbi.edu
*.nlc.edu
*.nlpbc.edu
*.nls.edu
*.nltcc.edu
*.nlts.edu
*.nlu.edu
*.nluconted.edu
*.nmc.edu
*.nmcc.edu
*.nmcnet.edu
*.nmcth.edu
*.nmhu.edu
*.nmims.edu
*.nmjc.edu
*.nmmi.edu
*.nmsu.edu
*.nmsua.edu
*.nmt.edu
*.nmti.edu
*.nmu.edu
*.nnmc.edu
*.nnu.edu
*.noao.edu
*.nobles.edu
*.nobts.edu
*.noc.edu
*.nocccd.edu
*.noce.edu
*.noctrl.edu
*.nodak.edu
*.noirlab.edu
*.nols.edu
*.nomenglobal.edu
*.normandale.edu
*.north-ok.edu
*.northampton.edu

*.northark.edu
*.northcoast.edu
*.northeast.edu
*.northern.edu
*.northgatech.edu
*.northland.edu
*.northpark.edu
*.northpoint.edu
*.northshore.edu
*.northsouth.edu
*.northwell.edu
*.northwest.edu
*.northwestms.edu
*.northwestu.edu
*.northwood.edu
*.norwalk.edu
*.norwalkcc.edu
*.norwich.edu
*.nosd.edu
*.nossi.edu
*.notredame.edu
*.nova.edu
*.novominsk.edu
*.novus.edu
*.np.edu
*.npaci.edu
*.npc.edu
*.npcc.edu
*.npcollege.edu
*.npi.edu
*.nps.edu
*.npti.edu
*.nptiflorida.edu
*.nptiohio.edu
*.npu.edu
*.nr.edu
*.nrait.edu
*.nrao.edu
*.nrc.edu
*.nsa.edu
*.nsc.edu
*.nscc.edu
*.nscee.edu
*.nscs.edu
*.nshs.edu
*.nsi.edu
*.nslijhs.edu
*.nsm.edu
*.nso.edu
*.nss.edu
*.nst.edu
*.nsu.edu
*.nsuba.edu
*.nsula.edu
*.nsuok.edu
*.ntc.edu
*.ntcc.edu
*.ntcmn.edu
*.nti.edu
*.ntid.edu
*.ntinow.edu
*.ntnu.edu
*.ntps.edu
*.nts.edu
*.ntt.edu
*.nttc.edu
*.ntts.edu
*.ntu.edu
*.nu.edu
*.nuc.edu
*.nuhs.edu
*.nujs.edu
*.numc.edu
*.nunez.edu
*.nunm.edu
*.nuol.edu
*.nur.edu
*.nus.edu
*.nusson.edu
*.nuvani.edu
*.nv.edu
*.nvcc.edu
*.nw.edu
*.nwa.edu
*.nwacc.edu
*.nwbs.edu
*.nwc.edu
*.nwcc.edu
*.nwciowa.edu
*.nwcollege.edu
*.nwcu.edu
*.nwculaw.edu
*.nwec.edu
*.nwfsc.edu
*.nwhealth.edu
*.nwhealthsci.edu
*.nwhsu.edu
*.nwic.edu
*.nwicc.edu
*.nwiht.edu
*.nwktc.edu
*.nwlett.edu
*.nwltc.edu
*.nwmissouri.edu
*.nwosu.edu
*.nws.edu
*.nwsc.edu
*.nwscc.edu
*.nwswb.edu
*.nwtc.edu
*.nwtech.edu
*.nwti.edu
*.nwts.edu
*.ny.edu
*.nyaa.edu
*.nyack.edu
*.nyadi.edu
*.nycc.edu
*.nycda.edu
*.nycenet.edu
*.nyci.edu
*.nycollege.edu
*.nycom.edu
*.nycpm.edu
*.nyctcm.edu
*.nyee.edu
*.nyfa.edu
*.nygc.edu
*.nyiad.edu
*.nyib.edu
*.nyicd.edu
*.nyieb.edu
*.nyim.edu
*.nyip.edu
*.nyit.edu
*.nyls.edu
*.nymahe.edu
*.nymc.edu
*.nymentor.edu
*.nysd.edu
*.nysid.edu
*.nyts.edu
*.nyu.edu
*.nyuivf.edu
*.oaa.edu
*.oak.edu
*.oakhills.edu
*.oakland.edu
*.oaklandcc.edu
*.oakpoint.edu
*.oakton.edu
*.oakvalley.edu
*.oakwood.edu
*.obcl.edu
*.oberlin.edu
*.obi.edu
*.obu.edu
*.oc.edu
*.ocac.edu
*.ocate.edu
*.ocb.edu
*.occ.edu
*.occc.edu
*.occidental.edu
*.ocean.edu
*.oceancorp.edu
*.ocemt.edu
*.ociw.edu
*.ocm.edu
*.ocom.edu
*.ocpm.edu
*.ocr.edu
*.octech.edu
*.ocu.edu
*.oda.edu
*.odessa.edu
*.odu.edu
*.oei.edu
*.oes.edu
*.oftc.edu
*.ogi.edu
*.ogleschool.edu
*.oglethorpe.edu
*.ogs.edu
*.oha.edu
*.ohio-state.edu
*.ohio.edu
*.ohiochrist.edu
*.ohiolink.edu
*.ohiostate.edu
*.ohiotech.edu
*.ohiou.edu
*.ohlone.edu
*.ohr.edu
*.ohrhameir.edu
*.ohsu.edu
*.oiah.edu
*.oikos.edu
*.oimt.edu
*.oipt.edu
*.oipwiozmfk.edu
*.oise.edu
*.oit.edu
*.ojc.edu
*.okbu.edu
*.okcu.edu
*.okstate.edu
*.okwu.edu
*.olc.edu
*.oldschool.edu
*.oldwestbury.edu
*.olemiss.edu
*.olhcc.edu
*.olin.edu
*.olivet.edu
*.ollusa.edu
*.ololcollege.edu
*.ols.edu
*.olympia.edu
*.olympian.edu
*.olympic.edu
*.omcc.edu
*.omega.edu
*.omi.edu
*.omicron.edu
*.omnis.edu
*.omnitech.edu
*.omsi.edu
*.omsk.edu
*.omw.edu
*.oneonta.edu
*.onlanguage.edu
*.onu.edu
*.open.edu
*.opi.edu
*.opmi.edu
*.opsu.edu
*.oralroberts.edu
*.oregoncoast.edu
*.oregonstate.edu
*.orion.edu
*.orleanstech.edu
*.orst.edu
*.ort.edu
*.ortn.edu
*.oru.edu
*.os.edu
*.osba.edu
*.osbc.edu
*.osc.edu
*.osrhe.edu
*.ossm.edu
*.ost.edu
*.ostm.edu
*.osu.edu
*.osucascades.edu
*.osuit.edu
*.osullivan.edu
*.osumc.edu
*.osuokc.edu
*.oswego.edu
*.otc.edu
*.otcweb.edu
*.otech.edu
*.otero.edu
*.oti-okc.edu
*.otis.edu
*.ottawa.edu
*.otterbein.edu
*.ou.edu
*.ouhsc.edu
*.oulu.edu
*.ous.edu
*.ovc.edu
*.ovct.edu
*.overbrook.edu
*.ovscdqraei.edu
*.ovu.edu
*.owatc.edu
*.owen.edu
*.owens.edu
*.owu.edu
*.oxford.edu
*.oxy.edu
*.ozarka.edu
*.ozarks.edu
*.p-i-a.edu
*.pac.edu
*.pacas.edu
*.pace.edu
*.paceacademy.edu
*.paceonline.edu
*.pacific.edu
*.pacifica.edu
*.pacificlife.edu
*.pacificoaks.edu
*.pacifictech.edu
*.pacificu.edu
*.packer.edu
*.pacollege.edu
*.pacrim.edu
*.pad.edu
*.paducahtech.edu
*.pafa.edu
*.pagunsmith.edu
*.paier.edu
*.paine.edu
*.pakaims.edu
*.palau.edu
*.palermo.edu
*.palladium.edu
*.palmer.edu
*.palmerwest.edu
*.palms.edu
*.palni.edu
*.palo-alto.edu
*.paloalto.edu
*.paloaltou.edu
*.palomar.edu
*.paloverde.edu
*.pamlicocc.edu
*.pams.edu
*.panam.edu
*.panola.edu
*.paragon.edu
*.paralegal.edu
*.paramount.edu
*.pardeerand.edu
*.pari.edu
*.paris.edu
*.parisjc.edu
*.park.edu
*.parker.edu
*.parkercc.edu
*.parkland.edu
*.parks.edu
*.parkwest.edu
*.paroba.edu
*.parsons.edu
*.pasadena.edu
*.passhe.edu
*.passion.edu
*.patgoins.edu
*.patten.edu
*.pau.edu
*.pauls.edu
*.paulsmiths.edu
*.payne.edu
*.pba.edu
*.pbacademy.edu
*.pbc.edu
*.pbcc.edu
*.pbcny.edu
*.pbcts.edu
*.pbisn.edu
*.pbrc.edu
*.pbsc.edu
*.pbu.edu
*.pc.edu
*.pcad.edu
*.pcage.edu
*.pcb.edu
*.pcc.edu
*.pccc.edu
*.pccenter.edu
*.pcci.edu
*.pccua.edu
*.pcd.edu
*.pcdi.edu
*.pcec.edu
*.pci.edu
*.pcicareer.edu
*.pcicareers.edu
*.pcicollege.edu
*.pcihealth.edu
*.pcim.edu
*.pcitraining.edu
*.pcj.edu
*.pcn.edu
*.pcom.edu
*.pcpro.edu
*.pcprofessor.edu
*.pcps.edu
*.pcsom.edu
*.pct.edu
*.pctc.edu
*.pcti.edu
*.pcu.edu
*.pdc.edu
*.pdi.edu
*.pdx.edu
*.pea.edu
*.peace.edu
*.peachnet.edu
*.pebblehills.edu
*.pec.edu
*.peel.edu
*.pei.edu
*.peirce.edu
*.pembroke.edu
*.pencol.edu
*.penfield.edu
*.penn.edu
*.pennco.edu
*.penncollege.edu
*.penncotech.edu
*.pennfoster.edu
*.pennwest.edu
*.penrose.edu
*.pepperdine.edu
*.peralta.edu
*.perbanas.edu
*.perelandra.edu
*.perrytech.edu
*.peru.edu
*.pes.edu
*.pespr.edu
*.pfeiffer.edu
*.pfi.edu
*.pfm.edu
*.pfrseminary.edu
*.pfw.edu
*.pgc.edu
*.pgcc.edu
*.pgi.edu
*.pgsp.edu
*.pgu.edu
*.pgups.edu
*.ph.edu
*.phagans.edu
*.phc.edu
*.phcc.edu
*.phdacademy.edu
*.philander.edu
*.philau.edu
*.phillips.edu
*.phoenix.edu
*.phoenixlaw.edu
*.photography.edu
*.phs.edu
*.phsc.edu
*.phssn.edu
*.phystech.edu
*.pi.edu
*.pia.edu
*.pianeta.edu
*.piartcenter.edu
*.piaschools.edu
*.pib.edu
*.pibc.edu
*.piberry.edu
*.pic.edu
*.piccollege.edu
*.picktwc.edu
*.picollege.edu
*.pict.edu
*.pictsctr.edu
*.pie.edu
*.piedmont.edu
*.piedmontcc.edu
*.piedmontu.edu
*.pierce.edu
*.piercelaw.edu
*.pierpont.edu
*.pihma.edu
*.piht.edu
*.pikespeak.edu
*.pillar.edu
*.pillsbury.edu
*.pima.edu
*.pims.edu
*.pinchot.edu
*.pine.edu
*.pinecrest.edu
*.pinetech.edu
*.pinewood.edu
*.pinnacle.edu
*.pioneerctc.edu
*.pioneertech.edu
*.pisd.edu
*.pit.edu
*.pitc.edu
*.pitt.edu
*.pittcc.edu
*.pittstate.edu
*.pitzer.edu
*.piu.edu
*.pivotpoint.edu
*.pjc.edu
*.pjscollege.edu
*.planet.edu
*.planwel.edu
*.platt.edu
*.plattsburgh.edu
*.plc.edu
*.plcc.edu
*.pli.edu
*.pllharvard.edu
*.pltc.edu
*.plts.edu
*.plu.edu
*.plymouth.edu
*.pm.edu
*.pmc.edu
*.pmi.edu
*.pmionline.edu
*.pmtc.edu
*.pmthelab.edu
*.pmthetemple.edu
*.pmtsgb.edu
*.pmu.edu
*.pnc.edu
*.pnca.edu
*.pns.edu
*.pnu.edu
*.pnuti.edu
*.pnw.edu
*.pnwcc.edu
*.pnwu.edu
*.pocatech.edu
*.point.edu
*.pointloma.edu
*.pointpark.edu
*.polaris.edu
*.polk.edu
*.poly.edu
*.pom.edu
*.pomona.edu
*.popac.edu
*.portergaud.edu
*.post.edu
*.postech.edu
*.potomac.edu
*.potsdam.edu
*.poynter.edu
*.ppcc.edu
*.ppg.edu
*.ppu.edu
*.pqc.edu
*.prairie.edu
*.praob.edu
*.pratt.edu
*.prattcc.edu
*.praxis.edu
*.prbc.edu
*.prbi.edu
*.prcc.edu
*.presby.edu
*.prescott.edu
*.presidio.edu
*.preston.edu
*.prgs.edu
*.prin.edu
*.princeton.edu
*.principia.edu
*.print.edu
*.printhusson.edu
*.prip.edu
*.privatfh-da.edu
*.processwork.edu
*.professorpc.edu
*.progressive.edu
*.prohands.edu
*.prometeu.edu
*.proskills.edu
*.protrain.edu
*.providence.edu
*.provo.edu
*.prts.edu
*.ps.edu
*.psb.edu
*.psba.edu
*.psc.edu
*.pscc.edu
*.psgtech.edu
*.psi.edu

*.psijax.edu
*.psjs.edu
*.psm.edu
*.psmt.edu
*.pson.edu
*.psow.edu
*.psp.edu
*.psr.edu
*.pstcc.edu
*.pstu.edu
*.psu.edu
*.psuca.edu
*.psy.edu
*.psychology.edu
*.psychosis.edu
*.ptc.edu
*.ptcc.edu
*.ptcollege.edu
*.ptec.edu
*.pti.edu
*.ptipr.edu
*.ptitraining.edu
*.ptmcaz.edu
*.pts.edu
*.ptsa.edu
*.ptsem.edu
*.ptseminary.edu
*.ptstulsa.edu
*.ptt.edu
*.pttc.edu
*.pua.edu
*.puc.edu
*.pucpr.edu
*.pueblocc.edu
*.pugetsound.edu
*.pulaskitech.edu
*.punahou.edu
*.pupr.edu
*.purchase.edu
*.purdue.edu
*.purduecal.edu
*.purduenw.edu
*.pust.edu
*.pvamu.edu
*.pvbi.edu
*.pvcc.edu
*.pwa.edu
*.pwcs.edu
*.pwscc.edu
*.qc.edu
*.qcc.edu
*.qmul.edu
*.qou.edu
*.qpcqirncnq.edu
*.qrc.edu
*.qschool.edu
*.qu.edu
*.quantic.edu
*.queencity.edu
*.queens.edu
*.quincy.edu
*.quinnipiac.edu
*.qvcc.edu
*.qvius.edu
*.racc.edu
*.race.edu
*.racs.edu
*.radcliffe.edu
*.radford.edu
*.rainyriver.edu
*.rajabhat.edu
*.rajagiri.edu
*.rajas.edu
*.ralc.edu
*.ram.edu
*.ramapo.edu
*.randolph.edu
*.ranken.edu
*.ranzco.edu
*.raritanval.edu
*.rasmussen.edu
*.rb.edu
*.rbc.edu
*.rc.edu
*.rca.edu
*.rcad.edu
*.rcbc.edu
*.rcbh.edu
*.rcc.edu
*.rccc.edu
*.rccd.edu
*.rcgc.edu
*.rci.edu
*.rcit.edu
*.rcpsc.edu
*.rcs.edu
*.rcsj.edu
*.rctc.edu
*.rdnational.edu
*.reach.edu
*.realtoru.edu
*.redeemer.edu
*.redlands.edu
*.redlandscc.edu
*.redstone.edu
*.redwoods.edu
*.reed.edu
*.reformation.edu
*.regency.edu
*.regent.edu
*.regents.edu
*.regis.edu
*.rei.edu
*.reinhardt.edu
*.reissdavis.edu
*.relay.edu
*.rensselaer.edu
*.res.edu
*.reseminary.edu
*.resu.edu
*.rets.edu
*.retstech.edu
*.reynolds.edu
*.rgec.edu
*.rghnet.edu
*.rgvcareers.edu
*.rgvcollege.edu
*.rh.edu
*.rhec.edu
*.rhodec.edu
*.rhodeisland.edu
*.rhodes.edu
*.rhodesstate.edu
*.ria.edu
*.riacs.edu
*.ric.edu
*.rice.edu
*.richland.edu
*.richmond.edu
*.richmondcc.edu
*.richmont.edu
*.rider.edu
*.ridge.edu
*.ridgewater.edu
*.ridley.edu
*.riets.edu
*.ringling.edu
*.rio.edu
*.riogrande.edu
*.riohondo.edu
*.riopc.edu
*.riosalado.edu
*.ripon.edu
*.risd.edu
*.rit.edu
*.ritindia.edu
*.ritz.edu
*.riverdale.edu
*.riverland.edu
*.riverside.edu
*.rivervalley.edu
*.rivier.edu
*.rjf.edu
*.rjj.edu
*.rjti.edu
*.rkc.edu
*.rknec.edu
*.rlc.edu
*.rli.edu
*.rlnc.edu
*.rm.edu
*.rma.edu
*.rmbc.edu
*.rmc.edu
*.rmcad.edu
*.rmcc.edu
*.rmcil.edu
*.rmu.edu
*.rmuohp.edu
*.rnu.edu
*.ro.edu
*.roanestate.edu
*.roanoke.edu
*.roberts.edu
*.robeson.edu
*.roch.edu
*.rochester.edu
*.rochesteru.edu
*.rockbridge.edu
*.rockefeller.edu
*.rockford.edu
*.rockhurst.edu
*.rockhursths.edu
*.rockies.edu
*.rocky.edu
*.rockymc.edu
*.rogersu.edu
*.roguecc.edu
*.rollins.edu
*.romania.edu
*.ronank12.edu
*.roosevelt.edu
*.rosary.edu
*.rose-hulman.edu
*.rose.edu
*.rosedale.edu
*.roseman.edu
*.rosemead.edu
*.rosemont.edu
*.rossmed.edu
*.rossonline.edu
*.rossu.edu
*.roswellpark.edu
*.rowan.edu
*.rowansj.edu
*.roxbury.edu
*.roytec.edu
*.rpcc.edu
*.rpi.edu
*.rpih.edu
*.rpslmc.edu
*.rpts.edu
*.rrc.edu
*.rrcc.edu
*.rrtc.edu
*.rsa.edu
*.rsc.edu
*.rsccd.edu
*.rsd.edu
*.rsht.edu
*.rsi.edu
*.rsiaz.edu
*.rst2.edu
*.rstc.edu
*.rstm.edu
*.rsu.edu
*.rsuonline.edu
*.rtc.edu
*.rts.edu
*.rtuvt.edu
*.ru.edu
*.rudaes.edu
*.ruiacollege.edu
*.runet.edu
*.runiv.edu
*.ruparel.edu
*.rush.edu
*.rushmore.edu
*.ruso.edu
*.rustcollege.edu
*.rutgers.edu
*.rutherford.edu
*.rvc.edu
*.rvs.edu
*.rvu.edu
*.rwanda.edu
*.rwc.edu
*.rwjuh.edu
*.rwjuhh.edu
*.rwtc.edu
*.rwu-prv.edu
*.rwu.edu
*.rwuonline.edu
*.ryokan.edu
*.rzyepcqzgz.edu
*.sa.edu
*.saa.edu
*.saaot.edu
*.saba.edu
*.sabanci.edu
*.sabanciuniv.edu
*.sac.edu
*.sachem.edu
*.sacn.edu
*.sacredheart.edu
*.saddleback.edu
*.sae.edu
*.safa.edu
*.safagava.edu
*.sagchip.edu
*.sage.edu
*.sagecollege.edu
*.sagrado.edu
*.sagu.edu
*.sai.edu
*.saic.edu
*.saice.edu
*.saintjoe.edu
*.saintleo.edu
*.saintmarys.edu
*.saintmikes.edu
*.saintpaul.edu
*.saintpauls.edu
*.saintpeters.edu
*.sais-jhu.edu
*.sal.edu
*.salem.edu
*.salemcc.edu
*.salemstate.edu
*.salemu.edu
*.salesianos.edu
*.salinatech.edu
*.salisbury.edu
*.salk.edu
*.salleurl.edu
*.sals.edu
*.salt.edu
*.salter.edu
*.salus.edu
*.salve.edu
*.sam.edu
*.samaritan.edu
*.samford.edu
*.sampsoncc.edu
*.samra.edu
*.samtech.edu
*.sandburg.edu
*.sandhills.edu
*.sandiego.edu
*.sanjac.edu
*.sanjuan.edu
*.sans.edu
*.santaclara.edu
*.santafe.edu
*.santarosa.edu
*.sanville.edu
*.sanz.edu
*.sapc.edu
*.sarasota.edu
*.sasin.edu
*.sastra.edu
*.sattler.edu
*.sau.edu
*.saumag.edu
*.sautech.edu
*.sawyer.edu
*.saxion.edu
*.saybrook.edu
*.sbac.edu
*.sbbcollege.edu
*.sbc.edu
*.sbcc.edu
*.sbccd.edu
*.sbci.edu
*.sbctc.edu
*.sbcuniv.edu
*.sbgi.edu
*.sbl.edu
*.sbmelville.edu
*.sbmri.edu
*.sbp.edu
*.sbs.edu
*.sbts.edu
*.sbu.edu
*.sbuniv.edu
*.sbw.edu
*.sc.edu
*.sc4.edu
*.scad.edu
*.scanlanhs.edu
*.scbc.edu
*.scbu.edu
*.scc.edu
*.sccbb.edu
*.sccc.edu
*.scccd.edu
*.scccwp.edu
*.scciowa.edu
*.sccky.edu
*.sccnc.edu
*.scco.edu
*.sccollege.edu
*.sccsc.edu
*.sce.edu
*.scf.edu
*.scfc.edu
*.schechter.edu
*.schev.edu
*.schi.edu
*.schiller.edu
*.schoolcraft.edu
*.schreiner.edu
*.schs.edu
*.sci.edu
*.sciarc.edu
*.science.edu
*.scindia.edu
*.scit.edu
*.scitech.edu
*.scitexas.edu
*.sckans.edu
*.scl.edu
*.scltc.edu
*.scmhrd.edu
*.scnm.edu
*.sco.edu
*.scope.edu
*.scotlewis.edu
*.scottlan.edu
*.scr.edu
*.scranton.edu
*.scripps.edu
*.scrippscol.edu
*.scs.edu
*.scsu.edu
*.sct.edu
*.sctc.edu
*.sctcc.edu
*.sctd.edu
*.sctech.edu
*.scti.edu
*.sctoday.edu
*.sctrain.edu
*.scu.edu
*.scuhs.edu
*.scusd.edu
*.scusoma.edu
*.sdbor.edu
*.sdc.edu
*.sdcc.edu
*.sdccd.edu
*.sdcce.edu
*.sdce.edu
*.sdcity.edu
*.sdean.edu
*.sdeyei-h.edu
*.sdgku.edu
*.sdi.edu
*.sdiae.edu
*.sdmcds.edu
*.sdmesa.edu
*.sdmiramar.edu
*.sdsc.edu
*.sdsmt.edu
*.sdstate.edu
*.sdsu.edu
*.sduis.edu
*.se-pr.edu
*.se.edu
*.sea.edu
*.seabury.edu
*.seal.edu
*.seark.edu
*.seattle.edu
*.seattleu.edu
*.sebc.edu
*.sebs.edu
*.sebts.edu
*.sec.edu
*.secon.edu
*.secwf.edu
*.sehcollege.edu
*.sejong.edu
*.sek.edu
*.sel.edu
*.selmau.edu
*.selnate.edu
*.selu.edu
*.seminary.edu
*.semo.edu
*.senmc.edu
*.sentara.edu
*.seoul.edu
*.sepr.edu
*.serrant.edu
*.seru.edu
*.ses.edu
*.sessions.edu
*.setai.edu
*.seteca.edu
*.seteho.edu
*.seti-inst.edu
*.setonhill.edu
*.seu.edu
*.sewanee.edu
*.sf.edu
*.sfai.edu
*.sfasu.edu
*.sfbc.edu
*.sfbu.edu
*.sfc.edu
*.sfcc.edu
*.sfccmo.edu
*.sfccnm.edu
*.sfcm.edu
*.sfcollege.edu
*.sfcpa.edu
*.sfd.edu
*.sfi.edu
*.sfiec.edu
*.sfls.edu
*.sfmccon.edu
*.sfs.edu
*.sfseminary.edu
*.sfsm.edu
*.sfsu.edu
*.sft.edu
*.sfts.edu
*.sftssc.edu
*.sfu.edu
*.sfusd.edu
*.sfvlaw.edu
*.sfwbc.edu
*.sgc.edu
*.sggs.edu
*.sgsc.edu
*.sgsl.edu
*.sgu.edu
*.sgus.edu
*.shadyside.edu
*.shafston.edu
*.sharif.edu
*.shasta.edu
*.shawnee.edu
*.shawneecc.edu
*.shawu.edu
*.shc.edu
*.shcp.edu
*.shctpt.edu
*.shearexcell.edu
*.sheffield.edu
*.shenandoah.edu
*.shepherd.edu
*.shepherds.edu
*.sheridan.edu
*.sherman.edu
*.shiloh.edu
*.shimer.edu
*.shin-ibs.edu
*.ship.edu
*.shm.edu
*.shms.edu
*.shoreline.edu
*.shorter.edu
*.showaboston.edu
*.shsst.edu
*.shst.edu
*.shsu.edu
*.shu.edu
*.shueyan.edu
*.si.edu
*.sia.edu
*.siam.edu
*.siba.edu
*.sibm.edu
*.sic.edu
*.siccm.edu
*.sictn.edu
*.sidwell.edu
*.sieam.edu
*.siena.edu
*.siescoms.edu
*.sigc.edu
*.signature.edu
*.siib.edu
*.silpakorn.edu
*.simc.edu
*.simmons.edu
*.simons-rock.edu
*.simpson.edu
*.simpsonu.edu
*.sims.edu
*.sinclair.edu
*.sinhgad.edu
*.sinica.edu
*.sintegleska.edu
*.siom.edu
*.sipi.edu
*.sir.edu
*.sirmvit.edu
*.sis.edu
*.siskiyous.edu
*.siss.edu
*.sit.edu
*.sittingbull.edu
*.siu.edu

*.siuc.edu
*.siue.edu
*.siuh.edu
*.siumed.edu
*.siusystem.edu
*.sjasc.edu
*.sjbtc.edu
*.sjc.edu
*.sjca.edu
*.sjcc.edu
*.sjcd.edu
*.sjcl.edu
*.sjcme.edu
*.sjcny.edu
*.sjcoe.edu
*.sjcs.edu
*.sjcsf.edu
*.sjctni.edu
*.sjeccd.edu
*.sjf.edu
*.sjfc.edu
*.sjfisher.edu
*.sjhcon.edu
*.sjny.edu
*.sjrcc.edu
*.sjrstate.edu
*.sjs.edu
*.sjson.edu
*.sjsu.edu
*.sju.edu
*.sjvc.edu
*.sjvconline.edu
*.sjvcs.edu
*.sjvdenver.edu
*.skagit.edu
*.skc.edu
*.skcca.edu
*.skema.edu
*.ski.edu
*.skidmore.edu
*.skinworks.edu
*.skku.edu
*.sksm.edu
*.sky.edu
*.skyctc.edu
*.skyline.edu
*.sl.edu
*.slc.edu
*.slcc.edu
*.slcconline.edu
*.slchc.edu
*.sli.edu
*.sls.edu
*.slts.edu
*.slu.edu
*.slucare.edu
*.sluh.edu
*.sm.edu
*.smac.edu
*.smat.edu
*.smc.edu
*.smcah.edu
*.smcc.edu
*.smccd.edu
*.smccme.edu
*.smcm.edu
*.smcollege.edu
*.smcsc.edu
*.smcvt.edu
*.smecollege.edu
*.smfa.edu
*.smhsomi.edu
*.smiha.edu
*.smith.edu
*.sms.edu
*.smsu.edu
*.smu.edu
*.smuhmts.edu
*.smumn.edu
*.smwc.edu
*.snai.edu
*.snc.edu
*.snead.edu
*.snes.edu
*.snesl.edu
*.snhu.edu
*.sno.edu
*.snow.edu
*.snu.edu
*.snybuf.edu
*.soa.edu
*.socalsem.edu
*.socc.edu
*.socccd.edu
*.sochi.edu
*.sof.edu
*.sofia.edu
*.soka.edu
*.sol.edu
*.solacc.edu
*.solano.edu
*.solex.edu
*.solexian.edu
*.sollers.edu
*.solvay.edu
*.soma.edu
*.somaiya.edu
*.somerset.edu
*.sonia.edu
*.sonoma.edu
*.sonoran.edu
*.sotc.edu
*.sotech.edu
*.sou.edu
*.south.edu
*.southark.edu
*.southbaylo.edu
*.southeast.edu
*.southeastmn.edu
*.southern.edu
*.southernct.edu
*.southernwv.edu
*.southgatech.edu
*.southhills.edu
*.southside.edu
*.southwest.edu
*.sowela.edu
*.spa.edu
*.space.edu
*.spacecoast.edu
*.spalding.edu
*.spartan.edu
*.spatech.edu
*.spb.edu
*.spc.edu
*.spcc.edu
*.spce.edu
*.spcollege.edu
*.spcu.edu
*.specshoward.edu
*.spelman.edu
*.spencerian.edu
*.spenta.edu
*.spertus.edu
*.spfldcol.edu
*.sph.edu
*.spjc.edu
*.spokane.edu
*.spots.edu
*.spring.edu
*.springarbor.edu
*.springfield.edu
*.springhouse.edu
*.sps.edu
*.spscc.edu
*.spst.edu
*.spsu.edu
*.sptseminary.edu
*.spu.edu
*.spuni.edu
*.spurgeon.edu
*.spuvvn.edu
*.src.edu
*.srcc.edu
*.srel.edu
*.srmc.edu
*.srmcps.edu
*.srmscet.edu
*.srtc.edu
*.srttu.edu
*.sru.edu
*.ss.edu
*.ssag.edu
*.ssakc.edu
*.ssbc.edu
*.ssc.edu
*.sscc.edu
*.sscms.edu
*.sscok.edu
*.sscr.edu
*.sse.edu
*.sseriga.edu
*.ssga.edu
*.ssh.edu
*.ssi.edu
*.ssist.edu
*.ssl.edu
*.ssoc.edu
*.ssu.edu
*.ssw.edu
*.st-albans.edu
*.st-aug.edu
*.stac.edu
*.stageone.edu
*.stamford.edu
*.stanbridge.edu
*.standrews.edu
*.stanford.edu
*.stanfordlaw.edu
*.stanfordu.edu
*.stanly.edu
*.stanton.edu
*.starcareer.edu
*.stark.edu
*.starkstate.edu
*.statetechmo.edu
*.staugustine.edu
*.stbernards.edu
*.stc.edu
*.stcacademy.edu
*.stcc.edu
*.stcecilia.edu
*.stchas.edu
*.stchris.edu
*.stcl.edu
*.stclaire.edu
*.stclements.edu
*.stealth.edu
*.stech.edu
*.stedwards.edu
*.steiner.edu
*.stenotech.edu
*.stenotype.edu
*.stephens.edu
*.sterling.edu
*.stetson.edu
*.steu.edu
*.stevens.edu
*.stevenson.edu
*.stfrancis.edu
*.stgeorges.edu
*.stgeorgesds.edu
*.stgregorys.edu
*.sti.edu
*.stikom.edu
*.stillman.edu
*.stitech.edu
*.stjames.edu
*.stjohns.edu
*.stjohnsem.edu
*.stjson.edu
*.stkate.edu
*.stlawrence.edu
*.stlawu.edu
*.stlcc.edu
*.stlcop.edu
*.stleo.edu
*.stluke.edu
*.stlukeuniv.edu
*.stmartin.edu
*.stmartins.edu
*.stmary.edu
*.stmarys-ca.edu
*.stmarys.edu
*.stmarysem.edu
*.stmarytx.edu
*.stmatthews.edu
*.stmichael.edu
*.stmichaels.edu
*.stmikes.edu
*.stmu.edu
*.stnersess.edu
*.stockton.edu
*.stolaf.edu
*.stone.edu
*.stonechild.edu
*.stonehill.edu
*.stonybrook.edu
*.storm.edu
*.stots.edu
*.stowers.edu
*.stpauls.edu
*.stpsu.edu
*.strasberg.edu
*.stratford.edu
*.strathmore.edu
*.strayer.edu
*.strita.edu
*.stritch.edu
*.strose.edu
*.sts.edu
*.stsci.edu
*.ststephens.edu
*.stsuots.edu
*.stthom.edu
*.stthomas.edu
*.stts.edu
*.stu.edu
*.stuart.edu
*.stuy.edu
*.stvincent.edu
*.stvt.edu
*.stvti.edu
*.su.edu
*.suagm.edu
*.subr.edu
*.success.edu
*.suffolk.edu
*.sui.edu
*.sulc.edu
*.sullivan.edu
*.sulross.edu
*.sum.edu
*.summit.edu
*.summit1.edu
*.summitcc.edu
*.summitoic.edu
*.summitu.edu
*.sunbridge.edu
*.suncoast.edu
*.sundbyberg.edu
*.sungkyul.edu
*.sunm.edu
*.suno.edu
*.sunstate.edu
*.sunteched.edu
*.suny.edu
*.sunyacc.edu
*.sunybroome.edu
*.sunycentral.edu
*.sunycgcc.edu
*.sunyct.edu
*.sunyempire.edu
*.sunyerie.edu
*.sunyit.edu
*.sunyjcc.edu
*.sunynassau.edu
*.sunynorth.edu
*.sunyocc.edu
*.sunyoccc.edu
*.sunyonline.edu
*.sunyopt.edu
*.sunyorange.edu
*.sunypoly.edu
*.sunypress.edu
*.sunysb.edu
*.sunysccc.edu
*.sunysuffolk.edu
*.sunytccc.edu
*.sunyulster.edu
*.sunywcc.edu
*.suomi.edu
*.suonline.edu
*.surry.edu
*.sus.edu
*.suscc.edu
*.susla.edu
*.susqu.edu
*.susquehanna.edu
*.sussex.edu
*.sust.edu
*.sustech.edu
*.suu.edu
*.suva.edu
*.sva.edu
*.svc.edu
*.svcc.edu
*.svdp.edu
*.svots.edu
*.svsu.edu
*.svu.edu
*.svuca.edu
*.sw.edu
*.swarthmore.edu
*.swatc.edu
*.swau.edu
*.swbts.edu
*.swc.edu
*.swcaz.edu
*.swcc.edu
*.swccd.edu
*.swcciowa.edu
*.swcenter.edu
*.swcollege.edu
*.swcu.edu
*.swfc.edu
*.swic.edu
*.swiha.edu
*.swinburne.edu
*.swiu.edu
*.swlaw.edu
*.swmed.edu
*.swmich.edu
*.swosu.edu
*.swri.edu
*.sws.edu
*.swsbs.edu
*.swtc.edu
*.swtech.edu
*.swtjc.edu
*.swu.edu
*.sxccal.edu
*.sxrtvu.edu
*.sxu.edu
*.sydenham.edu
*.sydney.edu
*.syr.edu
*.syracuse.edu
*.t-bird.edu
*.t-u.edu
*.ta.edu
*.taac.edu
*.tabor.edu
*.tabularasa.edu
*.tac.edu
*.tacomacc.edu
*.taft.edu
*.taftcollege.edu
*.taftu.edu
*.tahsenglish.edu
*.tai.edu
*.taikenku.edu
*.takming.edu
*.talbot.edu
*.taliesin.edu
*.talk.edu
*.talladega.edu
*.talmudicu.edu
*.tam.edu
*.tambcd.edu
*.tamhsc.edu
*.tamiu.edu
*.tamu.edu
*.tamuc.edu
*.tamucc.edu
*.tamuct.edu
*.tamug.edu
*.tamuk.edu
*.tamus.edu
*.tamusa.edu
*.tamut.edu
*.taofnj.edu
*.tarleton.edu
*.tatc.edu
*.tati.edu
*.taylor.edu
*.tayloru.edu
*.tbc.edu
*.tbc2day.edu
*.tbi.edu
*.tbiil.edu
*.tbr.edu
*.tbs.edu
*.tbu.edu
*.tc-japan.edu
*.tc.edu
*.tc3.edu
*.tca.edu
*.tcatathens.edu
*.tcatcrump.edu
*.tcatdickson.edu
*.tcatjackson.edu
*.tcatmemphis.edu
*.tcatnewbern.edu
*.tcatoneida.edu
*.tcatparis.edu
*.tcatpulaski.edu
*.tcatripley.edu
*.tcc.edu
*.tccd.edu
*.tcd.edu
*.tce.edu
*.tcgs.edu
*.tci.edu
*.tcicollege.edu
*.tcl.edu
*.tcm.edu
*.tcmc.edu
*.tcmch.edu
*.tcmi.edu
*.tcnj.edu
*.tcr.edu
*.tcsedsystem.edu
*.tcsg.edu
*.tctc.edu
*.tctcm.edu
*.tcu.edu
*.tcuglobal.edu
*.td.edu
*.tdds.edu
*.tea.edu
*.teach-now.edu
*.teccollege.edu
*.technical.edu
*.technologia.edu
*.techskills.edu
*.teco.edu
*.teds.edu
*.telecampus.edu
*.telematik.edu
*.telos.edu
*.telshe.edu
*.temple.edu
*.templejc.edu
*.tenet.edu
*.tenethealth.edu
*.tennessee.edu
*.terc.edu
*.terra.edu
*.tesc.edu
*.tesm.edu
*.tesst.edu
*.tesu.edu
*.teu.edu
*.texastcm.edu
*.texastech.edu
*.texshare.edu
*.tfa.edu
*.tfc.edu
*.tgif.edu
*.tgouwp.edu
*.tgsa.edu
*.th.edu
*.thapar.edu
*.the-bac.edu
*.thecoo.edu
*.thegateway.edu
*.thekings.edu
*.themodern.edu
*.thencc.edu
*.thenicc.edu
*.theology.edu
*.theoxford.edu
*.theseus.edu
*.thewoods.edu
*.thiel.edu
*.thomas.edu
*.thomasmore.edu
*.thomasu.edu
*.thompson.edu
*.thor.edu
*.threerivers.edu
*.ths.edu
*.thsu.edu
*.thunderbird.edu
*.ti.edu
*.tia.edu
*.tias.edu
*.tiasnimbas.edu
*.tiba.edu
*.tic.edu
*.ticas.edu
*.ticifl.edu
*.ticino.edu
*.tiep.edu
*.tiepatlamar.edu
*.tiffin.edu
*.tilburg.edu
*.timi.edu
*.tis.edu
*.tisc.edu
*.tisoh.edu
*.tiss.edu
*.titanium.edu
*.tiu.edu
*.tiua.edu
*.tjc.edu
*.tjhsst.edu
*.tjsl.edu
*.tju.edu
*.tkc.edu
*.tku.edu
*.tlbc.edu
*.tlc-corp.edu
*.tlc.edu
*.tlconline.edu
*.tlhu.edu
*.tls.edu
*.tlsohio.edu
*.tlu.edu
*.tm.edu
*.tmc.edu
*.tmcc.edu
*.tmcenter.edu
*.tmi.edu
*.tms.edu

*.tn.edu
*.tncc.edu
*.tnci.edu
*.tnstate.edu
*.tntech.edu
*.tntemple.edu
*.tnu.edu
*.tnwesleyan.edu
*.toa.edu
*.tocani.edu
*.tocc.edu
*.toccoafalls.edu
*.toi.edu
*.tokai.edu
*.tolani.edu
*.toledo.edu
*.toniguy.edu
*.tooeletech.edu
*.torah.edu
*.toronto.edu
*.tougaloo.edu
*.toulouse.edu
*.touro.edu
*.tourolaw.edu
*.touromed.edu
*.tourou.edu
*.tourow.edu
*.towson.edu
*.tpc.edu
*.trainace.edu
*.trance.edu
*.transy.edu
*.tras.edu
*.travel.edu
*.traviss.edu
*.trbc.edu
*.trcc.edu
*.trcoa.edu
*.trenton.edu
*.trevecca.edu
*.tri-c.edu
*.tri-state.edu
*.tricountycc.edu
*.trident.edu
*.tridenttech.edu
*.trin.edu
*.trincoll.edu
*.trine.edu
*.trinity.edu
*.trinitydc.edu
*.trinitypr.edu
*.trinitysem.edu
*.tristatecos.edu
*.triton.edu
*.trnty.edu
*.trocaire.edu
*.troy.edu
*.trtc.edu
*.truett.edu
*.truman.edu
*.trumbull.edu
*.tsacfotny.edu
*.tsb.edu
*.tsbc.edu
*.tsbi.edu
*.tsbvi.edu
*.tsc.edu
*.tsca.edu
*.tsd.edu
*.tse.edu
*.tsec.edu
*.tshas.edu
*.tsihd.edu
*.tsinghua.edu
*.tsm.edu
*.tsmu.edu
*.tsoa.edu
*.tsob.edu
*.tsot.edu
*.tst.edu
*.tstc.edu
*.tsu.edu
*.tsulaw.edu
*.tsuniv.edu
*.tsus.edu
*.ttcathens.edu
*.ttcc.edu
*.ttccrump.edu
*.ttcdickson.edu
*.ttcharriman.edu
*.ttcjackson.edu
*.ttcmckenzie.edu
*.ttcmemphis.edu
*.ttcnewbern.edu
*.ttconeida.edu
*.ttcparis.edu
*.ttcpulaski.edu
*.ttcripley.edu
*.ttic.edu
*.ttioc.edu
*.ttu.edu
*.ttuhsc.edu
*.ttuhscep.edu
*.tu-varna.edu
*.tu.edu
*.tubc.edu
*.tuc.edu
*.tuck.edu
*.tufts.edu
*.tui.edu
*.tuiu.edu
*.tul.edu
*.tulane.edu
*.tulsacc.edu
*.tulsatech.edu
*.tum.edu
*.tunxis.edu
*.turing.edu
*.turistica.edu
*.tusculum.edu
*.tuskegee.edu
*.tutorhusson.edu
*.tuw.edu
*.tvc.edu
*.tvcc.edu
*.twc.edu
*.twcnet.edu
*.tws.edu
*.twsu.edu
*.twu.edu
*.txbarber.edu
*.txchiro.edu
*.txinstitute.edu
*.txst.edu
*.txstate.edu
*.txwes.edu
*.txwesleyan.edu
*.tyndale.edu
*.ua.edu
*.uaa.edu
*.uab.edu
*.uabmc.edu
*.uac.edu
*.uaccb.edu
*.uacch.edu
*.uaccm.edu
*.uada.edu
*.uaex.edu
*.uaf.edu
*.uafortsmith.edu
*.uafs.edu
*.uag.edu
*.uagc.edu
*.uagm.edu
*.uagrantham.edu
*.uah.edu
*.uaht.edu
*.uakron.edu
*.ualberta.edu
*.ualr.edu
*.uamont.edu
*.uams.edu
*.uanet.edu
*.uap-bd.edu
*.uapar.edu
*.uapb.edu
*.uaptc.edu
*.uark.edu
*.uarts.edu
*.uas.edu
*.uasb.edu
*.uasd.edu
*.uasw.edu
*.uasys.edu
*.uasystem.edu
*.uat.edu
*.uav.edu
*.ub.edu
*.ubaguio.edu
*.ubalt.edu
*.ubatc.edu
*.ubc.edu
*.ubcdenver.edu
*.ubi.edu
*.uboces.edu
*.ubonline.edu
*.ubtech.edu
*.uc.edu
*.uc3m.edu
*.uca.edu
*.ucaid.edu
*.ucalgary.edu
*.ucam.edu
*.ucan.edu
*.ucanr.edu
*.ucar.edu
*.ucat.edu
*.ucax.edu
*.ucb.edu
*.ucbcomedy.edu
*.ucberkeley.edu
*.ucblueash.edu
*.ucc.edu
*.uccaribe.edu
*.ucclermont.edu
*.uccr.edu
*.uccs.edu
*.uccv.edu
*.ucd.edu
*.ucdavis.edu
*.ucdc.edu
*.ucdenver.edu
*.ucdh.edu
*.ucea.edu
*.uceda.edu
*.ucedaschool.edu
*.ucenter.edu
*.uceou.edu
*.ucerc.edu
*.ucf.edu
*.uch.edu
*.ucha.edu
*.uchastings.edu
*.uchc.edu
*.uchealth.edu
*.uchicago.edu
*.uchospitals.edu
*.uchsc.edu
*.uci.edu
*.ucirvine.edu
*.ucjc.edu
*.uckac.edu
*.ucl.edu
*.ucla.edu
*.uclm.edu
*.uclondon.edu
*.ucm.edu
*.ucmerced.edu
*.ucmo.edu
*.ucmt.edu
*.ucne.edu
*.uco.edu
*.ucollege.edu
*.uconline.edu
*.uconn.edu
*.ucop.edu
*.ucpress.edu
*.ucr.edu
*.ucriverside.edu
*.ucs.edu
*.ucsandiego.edu
*.ucsb.edu
*.ucsc.edu
*.ucsd.edu
*.ucsf.edu
*.ucsgh.edu
*.ucsystem.edu
*.uctm.edu
*.ucwv.edu
*.ucy.edu
*.udallas.edu
*.udayton.edu
*.udc.edu
*.udel.edu
*.udelistmo.edu
*.udem.edu
*.udf.edu
*.udg.edu
*.udi.edu
*.udla.edu
*.udlap.edu
*.udmercy.edu
*.udo.edu
*.ue.edu
*.uei.edu
*.uemc.edu
*.uewm.edu
*.uf.edu
*.ufairfax.edu
*.ufl.edu
*.ufm.edu
*.uft.edu
*.uftl.edu
*.uga.edu
*.ugf.edu
*.ugm.edu
*.ugst.edu
*.uh.edu
*.uhaulu.edu
*.uhc.edu
*.uhcc.edu
*.uhcl.edu
*.uhcno.edu
*.uhd.edu
*.uhsp.edu
*.uhsson.edu
*.uhsystem.edu
*.uhv.edu
*.ui.edu
*.uib.edu
*.uic.edu
*.uidaho.edu
*.uie.edu
*.uillinois.edu
*.uindy.edu
*.uiowa.edu
*.uip.edu
*.uipr.edu
*.uis.edu
*.uit.edu
*.uiu.edu
*.uiuc.edu
*.uiw.edu
*.uiwtx.edu
*.uj.edu
*.ujmv.edu
*.ukc.edu
*.uksw.edu
*.uky.edu
*.ula.edu
*.ulh.edu
*.ull.edu
*.ulm.edu
*.uls.edu
*.ulster.edu
*.ulsystem.edu
*.ultrasound.edu
*.ulv.edu
*.uma.edu
*.umab.edu
*.umaine.edu
*.umary.edu
*.umaryland.edu
*.umass.edu
*.umassd.edu
*.umassglobal.edu
*.umasslowell.edu
*.umassmed.edu
*.umassp.edu
*.umb.edu
*.umbc.edu
*.umbi.edu
*.umc.edu
*.umcaz.edu
*.umces.edu
*.umcrookston.edu
*.umd.edu
*.umdearborn.edu
*.umdnj.edu
*.umes.edu
*.umetro.edu
*.umfk.edu
*.umflint.edu
*.umgc.edu
*.umh.edu
*.umhb.edu
*.umhelena.edu
*.umhs.edu
*.umi-mdn.edu
*.umi.edu
*.umiami.edu
*.umich.edu
*.umiss.edu
*.umkc.edu
*.uml.edu
*.umm.edu
*.ummc.edu
*.ummed.edu
*.umn.edu
*.umo.edu
*.umobile.edu
*.umontana.edu
*.umpi.edu
*.umpqua.edu
*.ums.edu
*.umsl.edu
*.umsmed.edu
*.umsonline.edu
*.umss.edu
*.umsystem.edu
*.umt.edu
*.umtweb.edu
*.umuc.edu
*.umw.edu
*.umwestern.edu
*.una.edu
*.unai.edu
*.unam.edu
*.unamsa.edu
*.unav.edu
*.unb.edu
*.unbc.edu
*.unc.edu
*.unca.edu
*.uncc.edu
*.uncecs.edu
*.uncfsu.edu
*.uncg.edu
*.uncm.edu
*.unco.edu
*.unconline.edu
*.uncor.edu
*.uncp.edu
*.uncsa.edu
*.uncw.edu
*.uncwil.edu
*.und.edu
*.underwood.edu
*.une.edu
*.unem.edu
*.unex.edu
*.unf.edu
*.ung.edu
*.unh.edu
*.uni-pr.edu
*.uni-ulm.edu
*.uni-wh.edu
*.uni.edu
*.uniacc.edu
*.uniandina.edu
*.uniben.edu
*.unica.edu
*.unicah.edu
*.unilatina.edu
*.unilid.edu
*.uniminuto.edu
*.uninet.edu
*.union-psce.edu
*.union.edu
*.unionky.edu
*.uniq.edu
*.unir.edu
*.unisg.edu
*.unit.edu
*.unitec.edu
*.unitech.edu
*.unitechta.edu
*.unitecpr.edu
*.united.edu
*.unity.edu
*.univalle.edu
*.univdhaka.edu
*.university.edu
*.univnorthco.edu
*.univor.edu
*.unix.edu
*.unk.edu
*.unl.edu
*.unlv.edu
*.unm.edu
*.unmc.edu
*.unnj.edu
*.uno.edu
*.unoh.edu
*.unomaha.edu
*.unr.edu
*.unsw.edu
*.unt.edu
*.untdallas.edu
*.unterstrass.edu
*.unthsc.edu
*.untsystem.edu
*.unu.edu
*.unva.edu
*.unw.edu
*.unwsp.edu
*.uoa.edu
*.uoc.edu
*.uofa.edu
*.uofac.edu
*.uofillinois.edu
*.uofk.edu
*.uofl.edu
*.uofn.edu
*.uofnkona.edu
*.uofs.edu
*.uofsa.edu
*.uog.edu
*.uokhsc.edu
*.uoknor.edu
*.uona.edu
*.uop.edu
*.uopeople.edu
*.uophx.edu
*.uopx.edu
*.uor.edu
*.uoregon.edu
*.uosc.edu
*.up.edu
*.up8.edu
*.upacifico.edu
*.upal.edu
*.upb.edu
*.upc.edu
*.upcea.edu
*.upcomillas.edu
*.upenn.edu
*.upf.edu
*.uph.edu
*.upi.edu
*.upike.edu
*.upmc.edu
*.upr.edu
*.upra.edu
*.uprag.edu
*.uprb.edu
*.uprc.edu
*.uprh.edu
*.uprm.edu
*.uprovidence.edu
*.uprp.edu
*.uprrp.edu
*.uprs.edu
*.uprutuado.edu
*.ups.edu
*.upsem.edu
*.upstate.edu
*.uqyilrucgi.edu
*.ur.edu
*.urao.edu
*.urbana.edu
*.urbaniana.edu
*.urbe.edu
*.urg.edu
*.urgrgcc.edu
*.uri.edu
*.urich.edu
*.url.edu
*.urmc.edu
*.ursinus.edu
*.ursuline.edu
*.uru.edu
*.us.edu
*.usa.edu
*.usac.edu
*.usafa.edu
*.usaim.edu
*.usal.edu
*.usao.edu
*.usap.edu
*.usask.edu
*.usat.edu
*.usawc.edu
*.usb.edu
*.usc.edu
*.usca.edu
*.uscb.edu
*.uscga.edu
*.usciences.edu
*.uscny.edu
*.uscsumter.edu
*.uscupstate.edu
*.usd.edu
*.usdc.edu
*.usek.edu
*.useoul.edu
*.usf.edu
*.usfca.edu
*.usfsm.edu
*.usfsp.edu
*.usg.edu
*.ushe.edu
*.usi.edu
*.usiouxfalls.edu
*.usip.edu
*.usiu.edu
*.usiuafrica.edu
*.usj.edu
*.usk.edu
*.usla.edu
*.usling.edu
*.usm.edu
*.usma.edu
*.usmcu.edu
*.usmd.edu
*.usmf.edu

*.usml.edu
*.usmma.edu
*.usmsxm.edu
*.usn.edu
*.usna.edu
*.usncc.edu
*.usnh.edu
*.usnwc.edu
*.uso.edu
*.uson.edu
*.usou.edu
*.usouthal.edu
*.usp.edu
*.usps.edu
*.usra.edu
*.ussa.edu
*.usson.edu
*.ust.edu
*.ustb.edu
*.ustc.edu
*.ustdts.edu
*.usu.edu
*.usueastern.edu
*.usuhs.edu
*.usv.edu
*.usw.edu
*.usyd.edu
*.usz.edu
*.ut.edu
*.uta.edu
*.utah.edu
*.utahcollege.edu
*.utahsbr.edu
*.utahtech.edu
*.utam.edu
*.utampa.edu
*.utb.edu
*.utbtsc.edu
*.utc.edu
*.utcpr.edu
*.utd.edu
*.utdallas.edu
*.utdl.edu
*.utdt.edu
*.utech.edu
*.utep.edu
*.utepsa.edu
*.utesa.edu
*.utexas.edu
*.uth.edu
*.uthct.edu
*.uthouston.edu
*.uthsc.edu
*.uthscsa.edu
*.uti.edu
*.utica.edu
*.uticaonline.edu
*.utk.edu
*.utlaw.edu
*.utm.edu
*.utmartin.edu
*.utmb.edu
*.utmck.edu
*.utoledo.edu
*.utpa.edu
*.utpanam.edu
*.utpb.edu
*.utrgv.edu
*.uts.edu
*.utsa.edu
*.utsi.edu
*.utsny.edu
*.utsnyc.edu
*.utsouthern.edu
*.utsw.edu
*.utsystem.edu
*.uttc.edu
*.uttyl.edu
*.uttyler.edu
*.utu.edu
*.utulsa.edu
*.utwente.edu
*.utx.edu
*.utyler.edu
*.uu.edu
*.uuc.edu
*.uus.edu
*.uva.edu
*.uvawise.edu
*.uvei.edu
*.uvf.edu
*.uvi.edu
*.uvic.edu
*.uvinstitute.edu
*.uvm.edu
*.uvmnet.edu
*.uvs.edu
*.uvsc.edu
*.uvt.edu
*.uvu.edu
*.uw-mil.edu
*.uw.edu
*.uwa.edu
*.uwaonline.edu
*.uwaterloo.edu
*.uwb.edu
*.uwc.edu
*.uwec.edu
*.uwest.edu
*.uwex.edu
*.uwf.edu
*.uwgb.edu
*.uwi.edu
*.uwivet.edu
*.uwla.edu
*.uwlax.edu
*.uwm.edu
*.uwo.edu
*.uwosh.edu
*.uwp.edu
*.uwplatt.edu
*.uwrf.edu
*.uws.edu
*.uwsa.edu
*.uwsp.edu
*.uwstout.edu
*.uwsuper.edu
*.uww.edu
*.uwyo.edu
*.uzh.edu
*.vacu.edu
*.vai.edu
*.vak12ed.edu
*.valdocco.edu
*.valdosta.edu
*.valencia.edu
*.valenciacc.edu
*.vallauri.edu
*.valley.edu
*.valleyforge.edu
*.valparaiso.edu
*.valpo.edu
*.valverde.edu
*.vanderbilt.edu
*.vandercook.edu
*.vanguard.edu
*.vanity.edu
*.vantage.edu
*.varc.edu
*.vasom.edu
*.vassar.edu
*.vatech.edu
*.vatican.edu
*.vatterott.edu
*.vaughn.edu
*.vbc.edu
*.vbts.edu
*.vc.edu
*.vcc.edu
*.vcccd.edu
*.vccs.edu
*.vcfa.edu
*.vcmc.edu
*.vcom.edu
*.vconline.edu
*.vcsu.edu
*.vct.edu
*.vcu.edu
*.vdci.edu
*.veda.edu
*.venango.edu
*.vendee.edu
*.vennard.edu
*.venturalaw.edu
*.vermontlaw.edu
*.ves.edu
*.vesalius.edu
*.vesit.edu
*.vfcc.edu
*.vfmac.edu
*.vfs.edu
*.vgcc.edu
*.vgi.edu
*.vhacademy.edu
*.vhcc.edu
*.vic.edu
*.vici.edu
*.victory.edu
*.view.edu
*.vif.edu
*.vill.edu
*.villa.edu
*.villalan.edu
*.villanova.edu
*.villanueva.edu
*.vims.edu
*.vinu.edu
*.vips.edu
*.virginia.edu
*.viridis.edu
*.virtual.edu
*.visible.edu
*.vision.edu
*.vista.edu
*.vit.edu
*.viterbo.edu
*.viu.edu
*.vivekananda.edu
*.vksc.edu
*.vmcad.edu
*.vmi.edu
*.vms.edu
*.vmslaw.edu
*.vni.edu
*.vogue.edu
*.volny.edu
*.volstate.edu
*.voorhees.edu
*.vpcc.edu
*.vs.edu
*.vsc.edu
*.vshd.edu
*.vst.edu
*.vsu.edu
*.vt.edu
*.vtc.edu
*.vtcsom.edu
*.vti.edu
*.vts.edu
*.vtu.edu
*.vu.edu
*.vuim.edu
*.vul.edu
*.vumc.edu
*.vuom.edu
*.vuu.edu
*.vvc.edu
*.vwc.edu
*.vwu.edu
*.vxdkynqdya.edu
*.wa.edu
*.wab.edu
*.waba.edu
*.wabash.edu
*.wacad.edu
*.wadecollege.edu
*.wadhams.edu
*.waena.edu
*.wagner.edu
*.waialae.edu
*.waikato.edu
*.wakeforest.edu
*.wakehealth.edu
*.waketech.edu
*.waldenu.edu
*.waldorf.edu
*.wallace.edu
*.wallawalla.edu
*.walsh.edu
*.walshcol.edu
*.warnborough.edu
*.warner.edu
*.warren.edu
*.wartburg.edu
*.warwick.edu
*.wasatch.edu
*.wascae.edu
*.washburn.edu
*.washburnlaw.edu
*.washcampus.edu
*.washcoll.edu
*.washington.edu
*.washingtonu.edu
*.washint.edu
*.washjeff.edu
*.washlaw.edu
*.washu.edu
*.watc.edu
*.watkins.edu
*.watumull.edu
*.wau.edu
*.waubonsee.edu
*.wavecollege.edu
*.waycross.edu
*.wayman.edu
*.wayne.edu
*.waynecc.edu
*.waynesburg.edu
*.wba.edu
*.wbcoll.edu
*.wbcollege.edu
*.wbcs.edu
*.wbi.edu
*.wbs.edu
*.wbts.edu
*.wbu.edu
*.wc.edu
*.wcbc.edu
*.wcc.edu
*.wccc.edu
*.wcccd.edu
*.wccgb.edu
*.wccmt.edu
*.wccnet.edu
*.wccs.edu
*.wcdc.edu
*.wcfs.edu
*.wci.edu
*.wciu.edu
*.wcjc.edu
*.wcmo.edu
*.wcr.edu
*.wcs.edu
*.wcscc.edu
*.wcslc.edu
*.wcsu.edu
*.wctc.edu
*.wcu.edu
*.wcui.edu
*.wcupa.edu
*.wdt.edu
*.weacademy.edu
*.webb.edu
*.webber.edu
*.webc.edu
*.webcourse.edu
*.weber.edu
*.webster.edu
*.websteruniv.edu
*.wednet.edu
*.weimar.edu
*.welch.edu
*.wellesley.edu
*.wells.edu
*.wellspring.edu
*.wentworth.edu
*.wesley.edu
*.wesleyan.edu
*.west-mec.edu
*.west.edu
*.westal.edu
*.westbrooku.edu
*.westcliff.edu
*.westconn.edu
*.westech.edu
*.western.edu
*.westernsem.edu
*.westerntc.edu
*.westerntech.edu
*.westernu.edu
*.westernyork.edu
*.westfield.edu
*.westga.edu
*.westgatech.edu
*.westlawn.edu
*.westliberty.edu
*.westminster.edu
*.westmont.edu
*.westpoint.edu
*.westshore.edu
*.westtech.edu
*.westtown.edu
*.westvalley.edu
*.westwood.edu
*.wetcc.edu
*.wexford.edu
*.wfi.edu
*.wfu.edu
*.wfubmc.edu
*.wfusm.edu
*.wfusom.edu
*.wgtc.edu
*.wgu-rhp.edu
*.wgu.edu
*.wharton.edu
*.whatcom.edu
*.whccd.edu
*.wheaton.edu
*.wheatonma.edu
*.whecn.edu
*.wheeling.edu
*.wheelock.edu
*.whitefield.edu
*.whitehead.edu
*.whitman.edu
*.whittier.edu
*.whitworth.edu
*.whoi.edu
*.whs.edu
*.whu.edu
*.wi.edu
*.wiche.edu
*.wichita.edu
*.widener.edu
*.wilberforce.edu
*.wildwood.edu
*.wileyc.edu
*.wilkes.edu
*.wilkescc.edu
*.willamette.edu
*.williams.edu
*.williamsbu.edu
*.williamson.edu
*.wilmington.edu
*.wilmu.edu
*.wilson.edu
*.wilsoncc.edu
*.wiltech.edu
*.windsor.edu
*.winebrenner.edu
*.wingate.edu
*.winona.edu
*.winsor.edu
*.winstonprep.edu
*.winthrop.edu
*.wintu.edu
*.wiregrass.edu
*.wis.edu
*.wisc.edu
*.wisconsin.edu
*.wishard.edu
*.wisr.edu
*.wit.edu
*.witc.edu
*.witcc.edu
*.wits.edu
*.wittenberg.edu
*.wiu-usa.edu
*.wiu.edu
*.wjst.edu
*.wju.edu
*.wkccd.edu
*.wku.edu
*.wlac.edu
*.wlbc.edu
*.wlc.edu
*.wlu.edu
*.wm.edu
*.wma.edu
*.wmcarey.edu
*.wmcc.edu
*.wme.edu
*.wmed.edu
*.wmi.edu
*.wmich.edu
*.wmitchell.edu
*.wmpenn.edu
*.wmrc.edu
*.wmrs.edu
*.wms.edu
*.wmu.edu
*.wnc.edu
*.wncc.edu
*.wne.edu
*.wnec.edu
*.wnmu.edu
*.woebc.edu
*.wofford.edu
*.wolford.edu
*.woli.edu
*.won.edu
*.wongu.edu
*.woodbury.edu
*.woodland.edu
*.woodstock.edu
*.woodward.edu
*.wooster.edu
*.worcester.edu
*.worchester.edu
*.wordoflife.edu
*.world.edu
*.worldu.edu
*.worldwide.edu
*.worsham.edu
*.worwic.edu
*.wosc.edu
*.wou.edu
*.wpcc.edu
*.wpec.edu
*.wpi.edu
*.wpms.edu
*.wpunj.edu
*.wqu.edu
*.wright.edu
*.wrightcc.edu
*.wrightgrad.edu
*.wrs.edu
*.ws.edu
*.wsac.edu
*.wsc.edu
*.wscal.edu
*.wscc.edu
*.wschiro.edu
*.wscn.edu
*.wskiz.edu
*.wsmda.edu
*.wspp.edu
*.wssu.edu
*.wsu.edu
*.wsulaw.edu
*.wsutech.edu
*.wsuwest.edu
*.wtamu.edu
*.wtbc.edu
*.wtc.edu
*.wtcsystem.edu
*.wti.edu
*.wts.edu
*.wtsva.edu
*.wtti.edu
*.wtu.edu
*.wu.edu
*.wub.edu
*.wust.edu
*.wustl.edu
*.wuv.edu
*.wvbc.edu
*.wvc.edu
*.wvctcs.edu
*.wvhepc.edu
*.wvjc.edu
*.wvm.edu
*.wvncc.edu
*.wvnet.edu
*.wvsctc.edu
*.wvsom.edu
*.wvstateu.edu
*.wvsu.edu
*.wvu.edu
*.wvup.edu
*.wvutech.edu
*.wvwc.edu
*.wwcc.edu
*.wwgdiqoulo.edu
*.wwrc.edu
*.wwtech.edu
*.wwu.edu
*.wwwhusson.edu
*.wy.edu
*.wycliffe.edu
*.wyith.edu
*.wyoming.edu
*.wyotech.edu
*.xavier.edu
*.xaviers.edu
*.xlri.edu
*.xu.edu
*.xula.edu
*.yacollege.edu
*.yale.edu
*.ybi.edu
*.ybm.edu
*.yc.edu
*.yccc.edu
*.yccd.edu
*.ycce.edu
*.ycp.edu
*.ydc.edu
*.yei.edu
*.yeshiva.edu
*.ygss.edu
*.ygzm.edu
*.yhc.edu
*.yjecjvybzx.edu
*.yks.edu
*.yna.edu
*.ynkqefvdhm.edu
*.yoec.edu
*.yok.edu
*.yomiuri.edu
*.yonsei.edu
*.york.edu
*.yorkce.edu
*.yorkcol.edu
*.yorktech.edu
*.yosan.edu

*.yosemite.edu
*.yoy.edu
*.yst.edu
*.ysu.edu
*.ytc.edu
*.ytcnj.edu
*.ytech.edu
*.yti.edu
*.ytt.edu
*.yu.edu
*.yubacollege.edu
*.yuin.edu
*.yust.edu
*.yv.edu
*.yvcc.edu
*.yyh.edu
*.yza.edu
*.yzl.edu
*.zamorano.edu
*.zanestate.edu
*.zaytuna.edu
*.zbc.edu
*.zenshiatsu.edu
*.zg-ort.edu
*.zion.edu
*.zmc.edu
*.zmi.edu
*.zoni.edu
*.zuvbpcqanl.ed

Mari Silje Samuelson: Vivaldi ‘Four seasons’ - Presto from summer
https://www.youtube.com/watch?v=nJTfG1MmMwQ&ab_channel=MariSamuelsen
7.7M Views