ECE 574 – Cluster Computing
Lecture 22

Vince Weaver

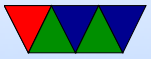
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

11 April 2023

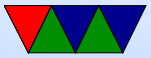
Announcements

- HW#9 will be posted soon
- Second Midterm will be Tuesday the 18th (week from today)
  Cumulative, though concentrating on recent material
- Project Status Report (see next)

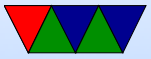


Status Report

- Provide some "related work" for your project. Do a literature search and find examples of other people who have done similar research. (Since this is a grad class)
  - Good skill to have for writing papers.
  - As they say, a month in the lab replaced by few hours in library
  - It's OK if you find something exactly the same. Reproducing work is important and under-utilized in academia.

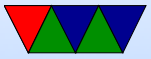


  - Even for masters degree it's OK if similar work is out there
  - PhD they expect you to have something unique
- I'd prefer if the references you find are books or academic papers, but (especially for programming projects) blog or website references are OK
  NOTE: don't pay for ACM/IEEE/etc papers. You can get them for free through UMaine library website
- Send report by e-mail, only one submission per group.
- State in one sentence a summary of your project
- Describe the hardware and software that you will be



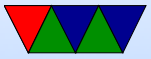
using.

- Have you made any progress on running code in such a setup
- The related work. Two references are fine for a single-person project, at least four if you have more than one person in your group.
- Will you be willing to volunteer to present early (Thursday 20th/25th/27th)



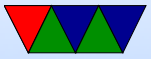
OpenCL HW#9 Notes

- Have to compile your kernel code
- Instead, can distribute binary only code. Why?
- Advanced: can set up queue and queued kernels only run when dependent ones are finished
- Calculating thread grids – did it for you?
- Trying to use integrated video on Haswell-ep? No intel integrated video on Xeon! It has a Matrox g200 chip from 1998!
- Using optimized intel CPU driver, interesting to see the



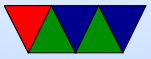
result. Using 8-bit char data on a GPU is not optimal, has to convert from int to float before doing calcs

- Questions about what things are faster, the CUDA manual chapter 5 has an optimization guide which is interesting to read.

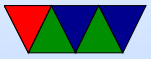


Big Data

- Until now compute or network bound systems
- What if we want to do a lot of disk/IO? Big Data?
- There are often trends in Computer Research
  - Takes a while to trickle down to UMaine, funny watching how administration will launch an initiative for the new "hot topic" after it has already peaked and is on the downswing elsewhere
  - Big Data was the previous hot thing, and sure enough UMaine just finalized its effort for it

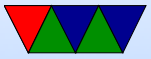


- The current big thing is AI. I thought that had peaked too, but this time the Chat-GPT stuff came in at the last minute as a bit of a curve-ball so maybe it will stick around longer

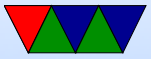


Big Data Examples

- Where is Data Used a lot?
  - Google
  - Worldwide LHC Computing Grid (WLCG) as of 2012
    25 petabytes of data/year (petabyte=1000 terabytes)
    300 GByte/s of data incoming
- Big Data is about big data sets (Terabytes?) and trying to get useful information out of them. Data Mining.
- "Big Data: Astronomical or Genomical": PLos Biology, 7 July 2015. (as per IEEE Spectrum

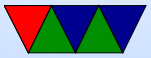


December 2015). Twitter: 1-17PB/year. Astronomy, 1,000PB/year, YouTube 1,000-2,000 PB/year, Genomics 2,000-40,000PB/year



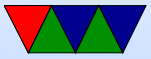
Big Data Challenges

- Data capture
- Data storage
- Data analysis / Visualization
- Data search / Querying
- Data sharing
- Data transfer
- Updating
- Information privacy (?) afterthought?



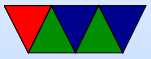
the Big Data Vs

- Volume – quantity, terabytes? Petabytes?
- Variety – more than just lists of numbers
- Velocity – speed the data is generated
- Veracity – GIGO (garbage in/garbage out)
- Value – Usefulness
- Variability –

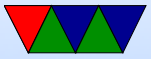


Big Data

- A buzzword?
- How big is big?
- Terabytes?
- Too big for one machine?
- In general if fits in RAM ($< 32GB$) or fits on disk ($< 10TB$) you are better off just using a database or similar
- Once it won't fit on one machine, and you want to use a cluster, things get complicated.



- Key idea is to *move computation to the data*, rather than vice-versa

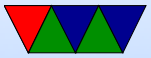


Big data challenges/astronomy

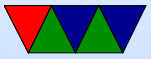
- https://www.computerworld.com/article/2972251/massive-telescope-array-aims-for-black-hole-gets-gu

  html

- Black hole "picture"
- From radio-wave interferometry
- Telescopes scattered all over world, including Antarctica
- Hard drives fail on mountain tops! (not enough air) use helium-filled ones instead
- Over 5 days, each telescope collected 900TB of data
- 1000-2000 hard drives, about 9PB



- How data sent? Hard-drives shipped to Massachusetts
- Had to wait for spring in Antarctica to ship out those
- 800 core cluster to analyze



Types of Storage

- Hard Disks
  - spinning rust – can be slow latency wise
  - SSD – faster, why?
  - Traditional vs Advanced features

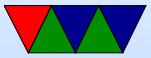
    Shingled (SMR) Disks

    Perpendicular (PMR) Disks

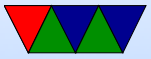
    `https://www.youtube.com/watch?v=xb_PyKuI7II`

    Helium

    Caches

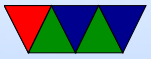


- Other flash / SD cards
- Memristors/Phase-Change/Optane/XPoint Non-volatile RAM
- Tape – robot tape libraries, etc
- Optical – CD/DVD/Blueray



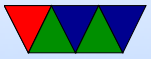
RAID

- Redundant Array of (Independent / Inexpensive) Disks
- Patterson Gibson and Katz 1987: replace expensive mainframe disks with arrays of relatively cheap desktop drives
- RAID0: striping, spreading across multiple disks, can increase performance, increases size of disk, bad things happen if one drive fails
- RAID1: mirroring – same thing written to both drives can increase performance as either drive can answer

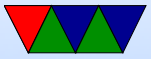


request

- RAID2: hamming code, each bit on separate drive. Not really used.
- RAID3: byte-level striping with parity. not common
- RAID4: block-level striping with dedicated parity.
- RAID5: block-level striping with distributed parity.
  can handle failure of single disk, can rebuild based on parity.
  Not recommended, as you have to read entirety of all other disks to rebuild, likely to fail other disks if all of same vintage

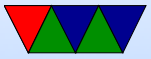


- RAID6: block-level striping with double parity. Recommended
- Hybrid: RAID10 = RAID1 + RAIDO
- Software vs Hardware
- Some filesystems include RAID like behavior: ZFS, GPFS, brfs, xfs

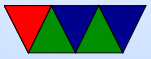


Non-RAID

- nearline storage – not offline but also not fully online
- MAID – Massive Array of Idle Drives
  - Write once, read occasionally.
  - Data that you want to save, but really don't access often so taking seconds to recall is OK.
  - What kind of data? Backups? Old Facebook pictures?
  - Old science data?



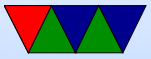
Traditional Ways to manage Data



Databases / RDBMs

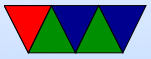
- Machines that store large amounts of data, often optimized for fast retrieval
- Databases / Relational Database Management System
- Relational databases: store rows of data, with a key. Each field has attribute.

  `Item, Name, Price, Color, Rating`



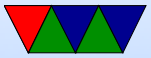
SQL (structured query language)

- Standard for querying relational database (this was a past "hot" computing topic)
- ```
  SELECT *
  FROM  Book
  WHERE price > 100.00
  ORDER BY title;
  ```
- Consistency?


  ```

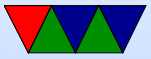
SQL Challenges

- Can have parallel and distributed databases too. It's more difficult with SQL
  - Replication – task runs, making sure all the various copies are kept in sync
  - Duplication – there is a master, and all the others are copies of the master. Users may only change master
- Main memory database – machines with 100TB of RAM?



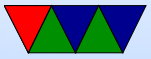
NoSQL Databases

- Scale-out architecture
  Can increase performance by adding nodes (rather than by upgrading single machine)
- Can store unstructured data (json, binary, text, sparse)
  Doesn't have to map to rows
- Can spread out on other machines, into cloud



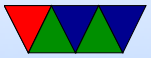
Cluster Filesystem

- Filesystem shared by multiple machines/nodes
- Can be centralized or distributed



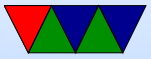
Shared-disk Filesystem

- Shared-disk filesystem – shares disk at block level
- SGI CXFS
  IBM GPFS
  Oracle Cluster Filesystem (OCFS)
  RedHat GFS
  Many Others
- RedHat GFS2
  Distributed Lock Manager



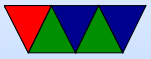
DAS – Direct Attached Storage

- typically how you hookup a hard-drive
- No network involved
- Relative low latency, but not distributed.
- Can be used as cache
- Can be exported for use as part of distributed filesystems



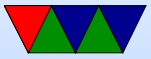
SAN – Storage Area Network

- (Don't confuse with NAS – network attached storage)
- A network that provides block-level access to data over a network and it appears to machines the same as local storage
- SAN often uses fibrechannel (fibre optics) but can also be over Ethernet



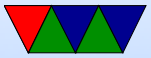
NAS – network attached storage

- Like a hard-drive you plug into the Ethernet but serves files (not disk blocks) usually by SMBFS (windows sharing), NFS, or similar
- NAS: appears to machine as a fileserver, SAN appears as a disk



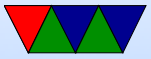
Network Storage Concerns

- QoS – quality of service. Limit bandwidth or latency so one user can't overly impact the rest
- Deduplication



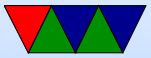
Cluster Storage

- Client-server vs Distributed
- Multiple servers?
- Distributed metadata – metadata spread out, not on one server
- Striping data across servers.
- Failover – if network splits, can you keep writing to files
- Disconnected mode.



Non-Distributed Network Filesystems

- NFS, SMBFS, Netware

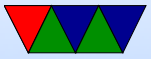


Distributed Filesystem Architectures

From *A Taxonomy and Survey on Distributed File Systems*

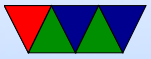
Can be any combination of the following

- Client Server – like NFS
- Cluster Based – single master with chunk servers
- Symmetric – peer to peer, all machines host some data with key-based lookup
- Asymmetric – separate metadata servers
- Parallel – data striped across multiple servers



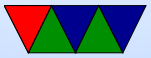
Stateless

Can you reboot server w/o client noticing? Lower overhead if server stateless because the server doesn't have to track every open file in the system



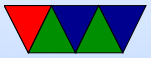
Synchronization/File Locking

- Multiple users writing to same file?
- Always synchronous?
- Same problems with consistency that you have with caches/memory

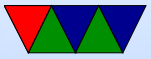


Consistency and Replication

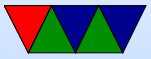
- Checksums to validate the data
- Caching – if you cache state of filesystem locally, how do you notice if other nodes have updated a file?



Failure Modes

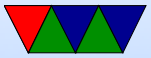


Security



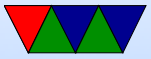
Distributed Filesystems

- Follow a network protocol, do not share block level access to disk
- Transparency is important. User doesn't need to know how it works underneath.
- Ceph, GFS, GlusterFS, Lustre, PVFS



Lustre

- "Linux Cluster"
- Complex ownership history
- Old article: `http://lwn.net/Articles/63536/`
- Used by many of the top 500 computers
  As of 2020, 77 of top 100 (rest IBM Spectrum scale)
  Frontier has 700TB 5GB/s Lustre filesystem
- Can handle tens of thousands of clients, tens of petabytes of data across hundreds of servers, and TB/s of I/O.
- One or more metadata servers: keep track of what files



exist, metadata, etc, locking, can load balance.

- One or more object storage servers
  Boxes of bits accessed by unique tag
- File can be "striped" across multiple storage servers and stream the file data in parallel
- Failure recovery. If node crashes, other nodes remember what it missed while down and help it recover to the proper state
- Distributed Locking
- Fast networking. Use RDMA when available.