ECE 574 – Cluster Computing
Lecture 2

Vince Weaver

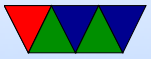
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

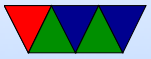
19 January 2023

Announcements

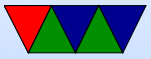
- HW#1 was due!
- A break on homeworks until next week.



Top500 List – November 2022

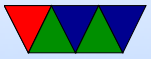


#	Name	Country	Arch /Proc	Cores	Max/Peak PFLOPS	Accel	Power kW
1	Frontier	US/ORNL	AMD EPYC	8,730,112	1.1k/1.7k	AMD Instinct	21MW
2	SC Fugaku	Japan/Riken	ARM64	7,630,848	442/537	N/A	30MW
3	LUMI	Finland	AMD EPYC	2,220,288	309/428	AMD Instinct	6MW
4	Leonardo	Italy	Intel ICL	1,463,616	174/255	NVD A100	6MW
5	Summit (IBM)	US/ORNL	IBM Power9	2,414,592	148/200	NVD Volta	10MW
6	Sierra (IBM)	US/LLNL	IBM Power9	1,572,480	94/125	NVD Volta	7MW
7	Sunway TaihuLight	China	Sunway	10,649,600	93/125	?	15.3MW
8	Perlmutter	US/LBNL	AMD EPYC	761,856	71/94	NVD A100	2.6MW
9	Selene	US/NVIDIA	AMD EPYC	555,520	63/79	NVD A100	2.6MW
10	Tianhe-2A	China	Intel IVB	4,981,760	61/101	MatrixDSP	18.5MW
11	Adastra	France	AMD EPYC	319,072	46/61	AMD Instinct	0.9MW
12	JUWLES Booster	Germany	AMD EPYC	449,280	44/71	NVD A100	1.7MW
13	HPC5	Italy	Intel CSL	669,760	35/51	NVD Tesla	2.25MW
14	Voyager-EUS2	US/Microsoft	AMD EPYC	253,440	30/40	A100	?MW
15	Setonix GPU	Australia	AMD EPYC	182,248	27/35	AMD Instinct	0.5MW
16	Discovery 5	US/Exxon	AMD EPYC	232,100	26/31	NVD A100	1MW
17	Polaris	US/Argonne	AMD EPYC	256,592	26/34	NVD A100	?MW
18	SSC-21	S. Korea/Smsng	AMD EPYC	204,160	25/32	NVD A100	?MW
19	Frontera	US/TACC	Intel CSL	448,448	23/38	N/A	? MW
20	CEA-HF	France	AMD EPYC	810,240	23/32	N/A	5 MW
21	Dammam-7	Saudi Arabia	Intel CSL	672,520	22/55	NVD V100	? MW
22	ABCI 2.0	Japan	Intel CSL	504,000	22/54	NVD A100	1.6MW
23	Wisteria	Japan	ARM A64FX	368,640	22/26	N/A	1.5MW
24	Marconi-100	Italy	Power9	347,776	21/29	NVD V100	1.5 MW
25	Chervonenkis	Russia	AMD EPYC	193,440	22/39	NVD A100	?MW

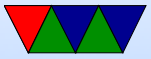


Top500 List Notes

- Left off my summary: RAM? Interconnect?
- Operating system? Mostly Linux these days
- Power: does this include cooling or not?
  Cost of power over lifetime of use is often higher than the cost to build it.
- Power comparison: small town? 1MW around 1000 homes? (this varies)
- How long does it take to run LINPACK? How much money does it cost to run LINPACK?

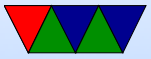


- Cost to run computer more than cost to build it?
- Finally hit exaflop
- Intel not as dominant. AMD EPYC everywhere, ARM also



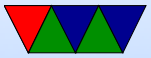
Notes on the Top500 BoF Video

- If you watched the whole thing...
- Frontier
  - 74 racks, 9.2PB RAM (half HB, half DDR4)
  - 90 miles of network cable
  - $600 million
  - Quiet, water cooled. Warm water (32C)
  - Trouble getting more thn 600PFLOPs, turned out to be linear-time thing in Cray MPI library
  - 3 hours to run Linpack, nodes keep failing when try to



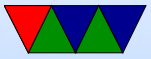
do run

- Non-linpack results, HPGC Frontier #2 only 14 PFLOPs
- Green 500, GFLOPs/W. Frontier much lower. Top was machine with first NVIDIA H100 (Hpper)
- Systems appearing more slowly on list, aging more before dropping off



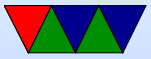
What goes into a top supercomputer?

- Commodity or custom
- Architecture: x86? SPARC? Power? ARM
  embedded vs high-speed?
- Memory
- Storage
  How much?
  Large hadron collider one petabyte of data every day
  Shared? If each node wants same data, do you need to replicate it, have a network filesystem, copy it around



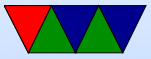
with jobs, etc? Cluster filesystems?

- Reliability. How long can it stay up without crashing?
  Can you checkpoint/restart jobs?
  Sequoia MTBF 1 day.
  Blue Waters 2 nodes failure per day.
  Titan MTBF less than 1 day
- Power / Cooling
  Big river nearby?
- Accelerator cards / Heterogeneous Systems
- Network
  How fast? Latency? Interconnect? (torus, cube,

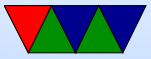


hypercube, etc)
Ethernet? Infiniband? Custom?

- Operating System
  Linux? Custom? If just doing FP, do you need overhead of an OS?
- Job submission software, Authentication
- Software – how to program?
  Too hard to program can doom you. A lot of interest in the Cell processor. Great performance if programmed well, but hard to do.
- Tools – software that can help you find performance

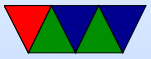


problems



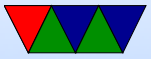
Other stuff

- Rmax vs Rpeak – Rmax is max measured, Rpeak is theoretical best
- HPL Linpack
  - Embarrassingly parallel linear algebra
  - Solves a (random) dense linear system in double precision (64 bits) arithmetic
- HP Conjugate gradient benchmark
  - More realistic? Does more memory access, more I/O bound.



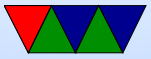
- #1 on list is Fugaku. 16PFLOPS CG wheras 442PFLOPS HPL
- Some things can move around, K-computer 18th in HPL but 3rd with CG

- Green 500

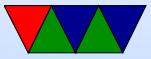


Historical Note

- From the November 2002 list, entry #332
- Location: Orono, ME
- Proc Arch: x86
- Proc Type: Pentium III, 1GHz
- Total cores: 416
- RMax/RPeak: 225/416 GFLOPS
- Power: ???
- Accelerators: None

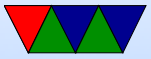


Introduction to Performance Analysis



What is Performance?

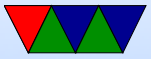
- Getting results as quickly as possible?
- Getting *correct* results as quickly as possible?
- What about Budget?
- What about Development Time?
- What about Hardware Usage?
- What about Power Consumption?



Motivation for HPC Optimization

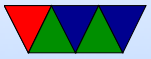
HPC environments are expensive:

- Procurement costs: $\sim\$40$ million
- Operational costs: $\sim\$5$ million/year
- Electricity costs: 1 MW / year $\sim\$1$ million
- Air Conditioning costs: ??

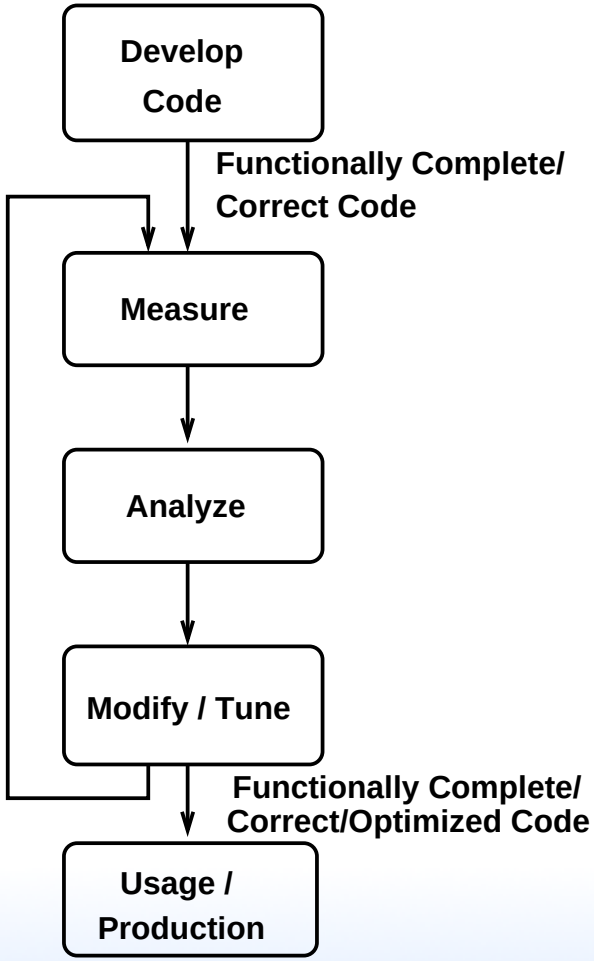


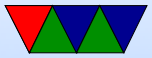
Know Your Limitation

- CPU Constrained
- Memory Constrained (Memory Wall)
- I/O Constrained
- Thermal Constrained
- Energy Constrained



Performance Optimization Cycle



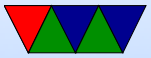


Wisdom from Knuth

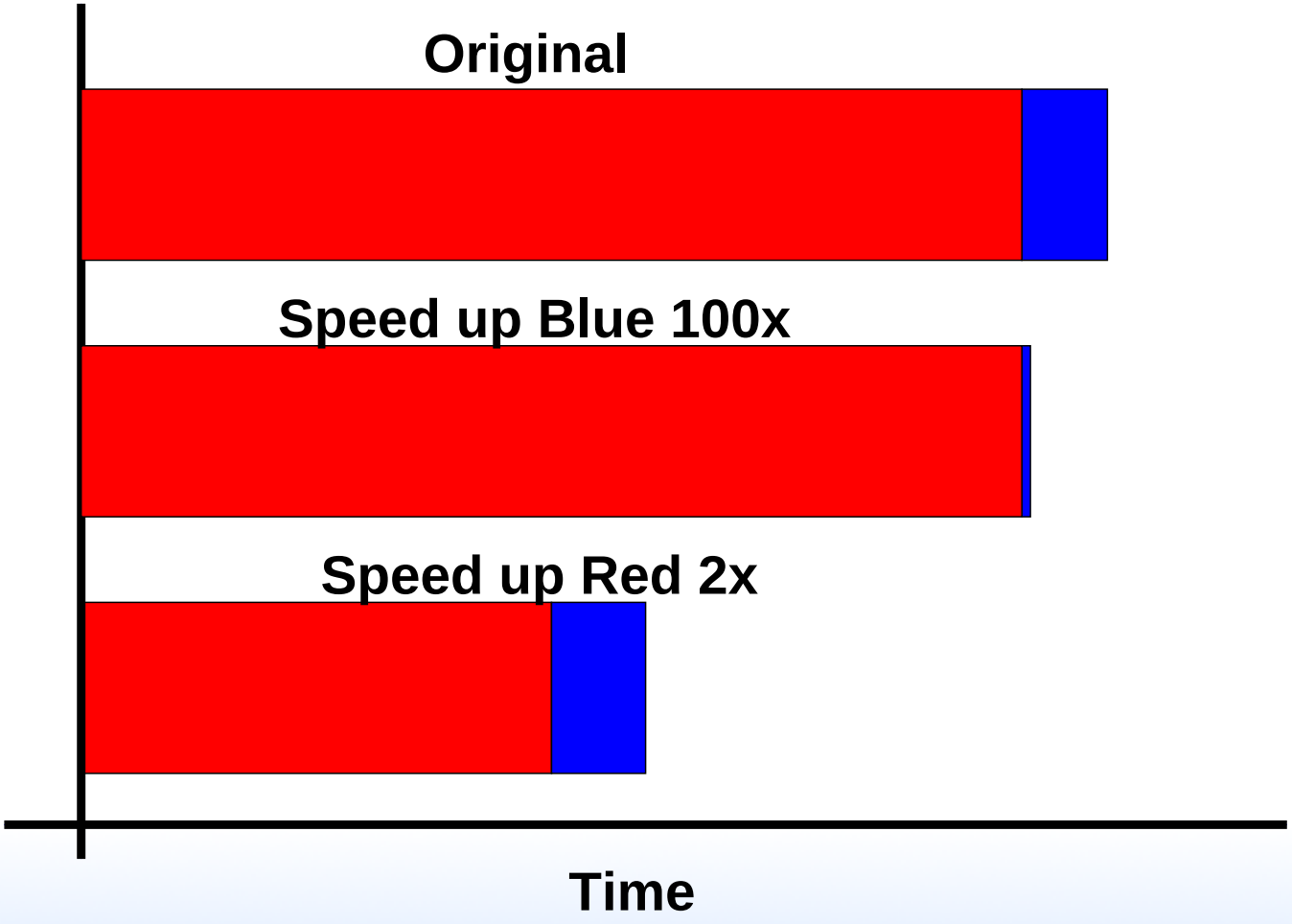
"We should forget about small efficiencies, say about 97% of the time:

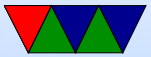
premature optimization is the root of all evil.

Yet we should not pass up our opportunities in that critical 3%. A good programmer will not be lulled into complacency by such reasoning, he will be wise to look carefully at the critical code; but only after that code has been identified" — Donald Knuth



Amdahl's Law



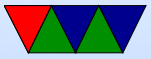


Speedup

- Speedup is the improvement in latency (time to run)

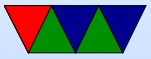
$$S = \frac{t_{old}}{t_{new}}$$

So if originally took 10s, new took 5s, then speedup=2.



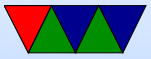
Scalability

- How a workload behaves as more processors are added
- Parallel efficiency: $E_p = \frac{S_p}{p} = \frac{T_s}{pT_p}$
  p=number of processes (threads)
  T_s is execution time of serial code
  T_p is execution time with p processes
- Linear scaling, ideal: $S_p = p$
- Super-linear scaling – possible but unusual



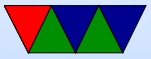
Strong vs Weak Scaling

- Strong Scaling –for fixed program size, how does adding more processors help
- Weak Scaling – how does adding processors help with the same per-processor workload



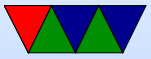
Strong Scaling

- Have a problem of a certain size, want it to get done faster.
- Ideally with problem size N, with 2 cores it runs twice as fast as with 1 core (linear speedup)
- Often processor bound; adding more processing helps, as communication doesn't dominate
- Hard to achieve for large number of nodes, as many



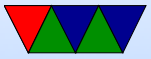
algorithms communication costs get larger the more nodes involved

- Amdahl's Law limits things, as more cores don't help serial code
- Strong scaling efficiency: t1 / (N * tN) * 100%
- Improve by throwing CPUs at the problem.

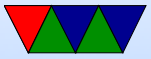


Weak Scaling

- Have a problem, want to increase problem size without slowing down.
- Ideally with problem size N with 1 core, a problem of size 2*n just as fast with 2 cores.
- Often memory or communication bound.
- Gustafson's Law (rough paraphrase)
  No matter how much you parallelize your code, there will be serial sections that just can't be made parallel

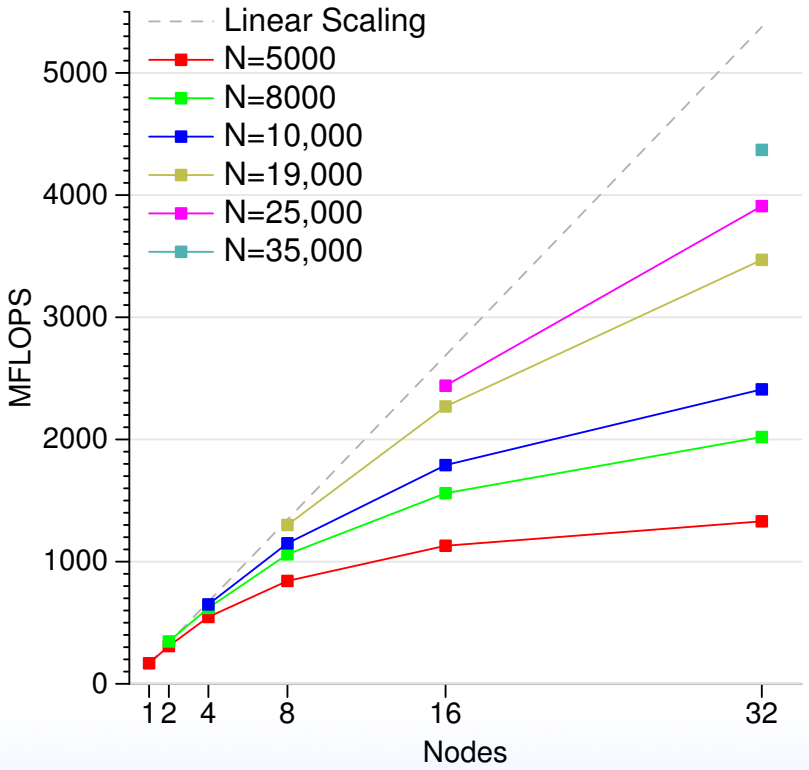


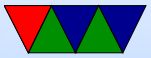
- Weak scaling efficiency: (t1 / tN) * 100%
- Improve by adding memory, or improving communication?



Scaling Example

LINPACK on Rasp-pi cluster. What kind of scaling is here?





Weak scaling. To get linear speedup need to increase problem size.

If it were strong scaling, the individual colored lines would increase rather than dropping off.