ECE 574 – Cluster Computing
Lecture 2

Vince Weaver

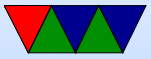
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

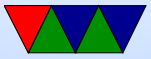
28 January 2021

Announcements

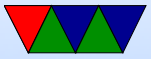
- HW#1 was due!
- A break on homeworks until next week.



Top500 List – November 2020

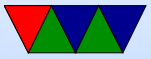


#	Name	Country	Arch /Proc	Cores	Max/Peak PFLOPS	Accel	Power kW
1	Supercomputer Fugaku	Japan/Riken	ARM64	7,630,848	442/537	N/A Volta	30MW
2	Summit (IBM)	US/ORNL	Power9	2,414,592	148/200	NVD Volta	10MW
3	Sierra (IBM)	US/LLNL	Power9	1,572,480	94/125	NVD Volta	7MW
4	TaihuLight	China	Sunway	10,649,600	93/125	?	15.3MW
5	Selene	US/NVIDIA	AMD EPYC	555,520	63/79	NVD A100	2.6MW
6	Tianhe-2A	China	x86/IVB	4,981,760	61/101	MatrixDSP	18.5MW
7	JUWLES Booster	Germany	AMD EPYC	449,280	44/71	NVD A100	1.7MW
8	HPC5	Italy	x86/CSL	669,760	35/51	NVD Tesla	2.25MW
9	Frontera	US/TACC	x86/CSL	448,448	23/38	N/A	? MW
10	Dammam-7	Saudi Arabia	x86/CSL	672,520	22/55	NVD V100	? MW
11	Marconi-100	Italy	Power9	347,776	21/29	NVD V100	1.5 MW
12	Piz Daint (Cray)	Swiss	x86/SNB	387,872	21/27	NVD Tesla	2,384
13	Trinity (Cray)	US/LANL	x86/HSW	979,072	20/158	XeonPhi	7,578
14	ABCI (Fujitsu)	Japan	x86/SKL	391,680	20/33	NVD Tesla	1,649
15	SuperMUC-NG	Germany	x86/SKL	305,856	19/26	?	?
16	Hawk	Germany	AMD EPYC	698,880	19/25	?	2.9MW
17	Lassen (IBM)	USA/LLNL	Power9	288,288	18/23	NVD Tesla	?
18	Pangea III	France	Power9	291,024	17/25	NVD Volta	1.3MW
19	TOKI-SORA	Japan	ARM64	276,480	16/19	?	?MW
20	Cori (Cray)	USA/LBNL	x86/HSW	622,336	14/27	Xeon Phi	4MW

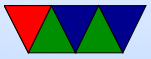


Top500 List Notes

- Linked to the BoF video in homework
- Left off my summary: RAM? (#1 is 5PB) Interconnect?
- 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?

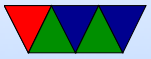


- Lots of turnover since last time I taught the class?
- Operating system. Cost to run computer more than cost to build it?
- Tiahne-2 was Xeon Phi, but US banned Intel from exporting anymore, so upgraded and using own custom DSP boards now.
- Getting closer to exaflop, was goal for 2020.
- Intel not as dominant. ARM at top.



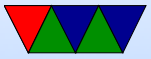
What goes into a top supercomputer?

- 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



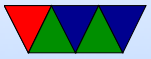
with jobs, etc? Cluster filesystems?

- 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,



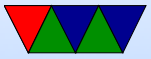
hypercube, etc)
Ethernet? Infiniband? Custom?

- 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



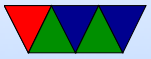
Other stuff

- 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.



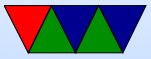
- #1 on list is Fugaku. 16PFLOPS CG wheras 442PFLOPS HPL
- Some things can move around, K-computer 18th in HPL but 3rd with CG

- Green 500

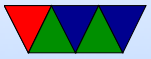


Historical Note

- 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

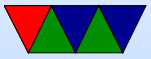


Introduction to Performance Analysis



What is Performance?

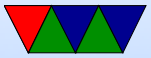
- Getting results as quickly as possible?
- Getting *correct* results as quickly as possible?
- What about Budget?
- What about Development Time?
- What about Hardware Usage?
- What about Power Consumption?



Motivation for HPC Optimization

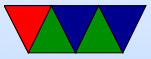
HPC environments are expensive:

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

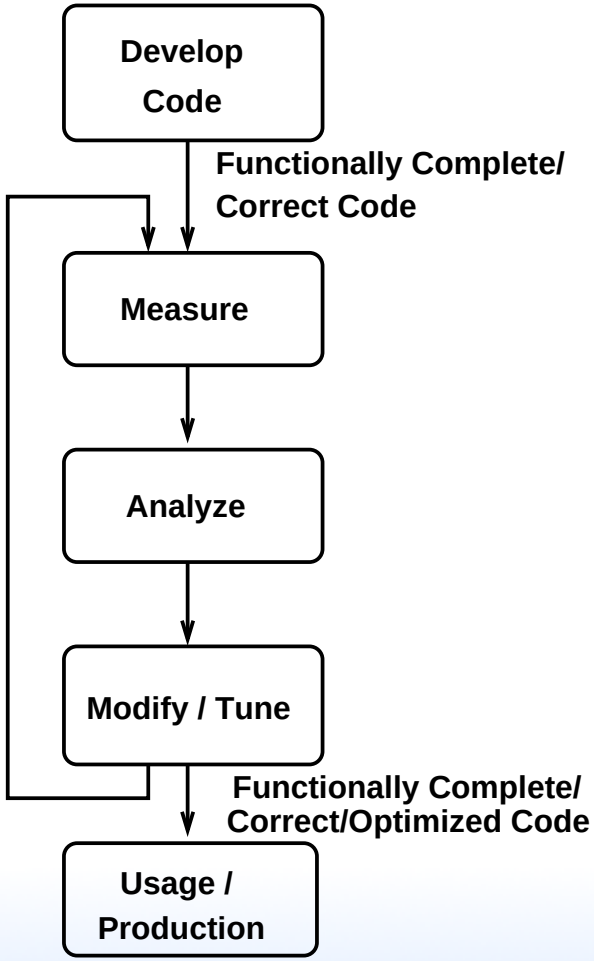


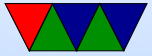
Know Your Limitation

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



Performance Optimization Cycle



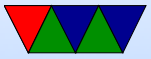


Wisdom from Knuth

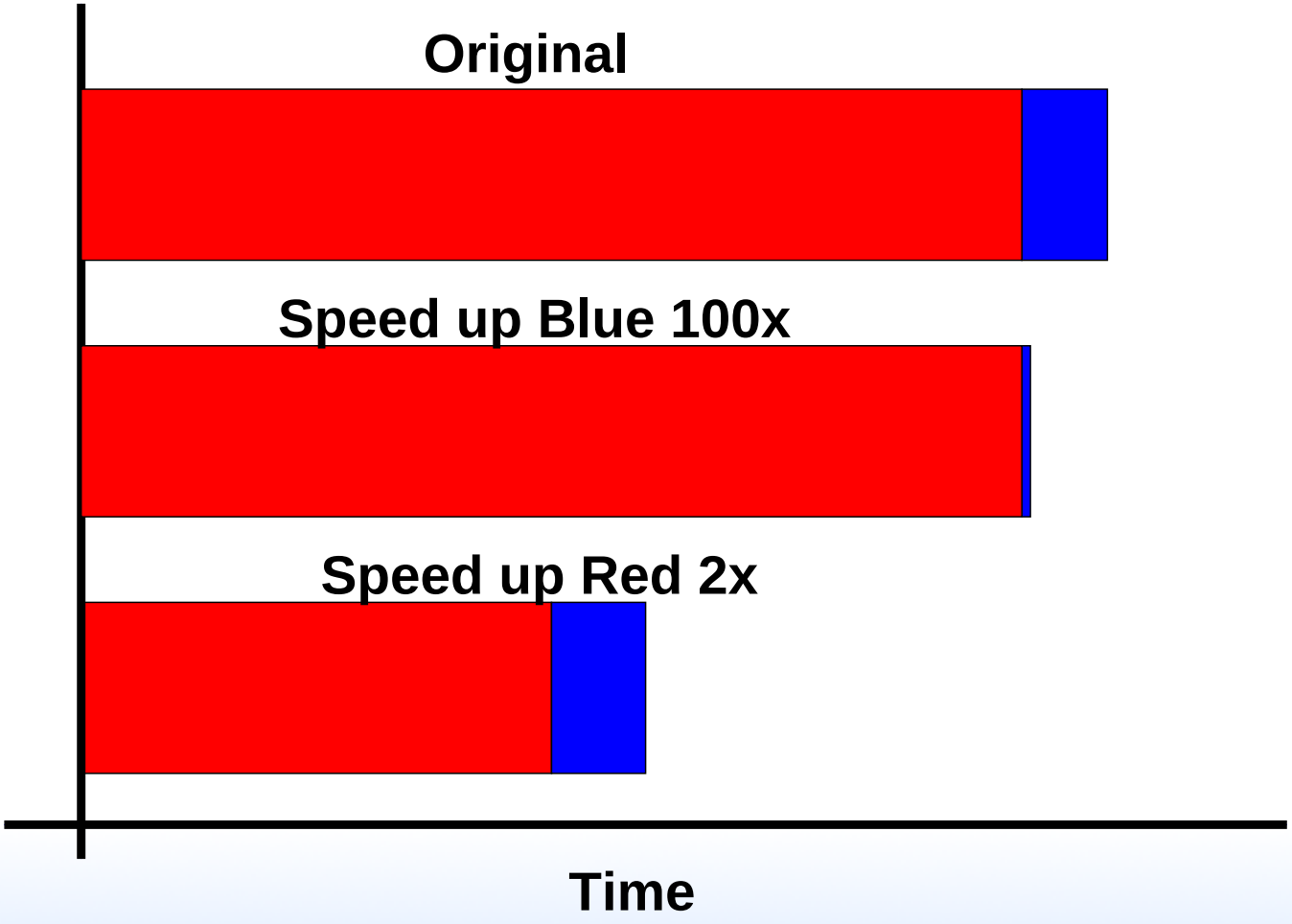
"We should forget about small efficiencies, say about 97% of the time:

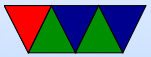
premature optimization is the root of all evil.

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



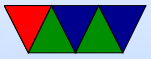


Speedup

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

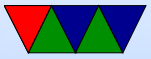
$$S = \frac{t_{old}}{t_{new}}$$

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



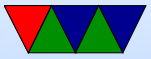
Scalability

- 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



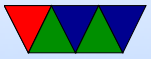
Strong vs Weak Scaling

- 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



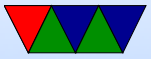
Strong Scaling

- 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



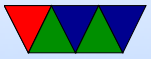
algorithms communication costs get larger the more nodes involved

- 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.

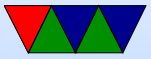


Weak Scaling

- 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

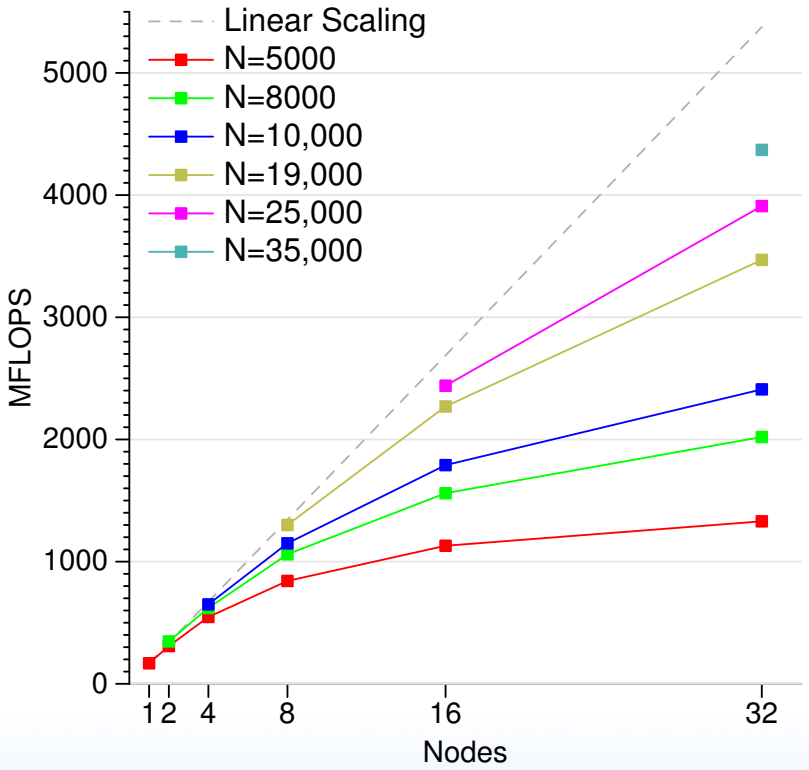


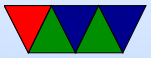
- Weak scaling efficiency: (t1 / tN) * 100%
- Improve by adding memory, or improving communication?



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.