COMP/CS 605: Intro to Parallel Computing
Lecture 01: Parallel Computing Overview

Mary Thomas

Department of Computer Science
Computational Science Research Center (CSRC)
San Diego State University (SDSU)

Posted: 01/19/17
Updated: 01/19/17

Table of Contents

Course Updates

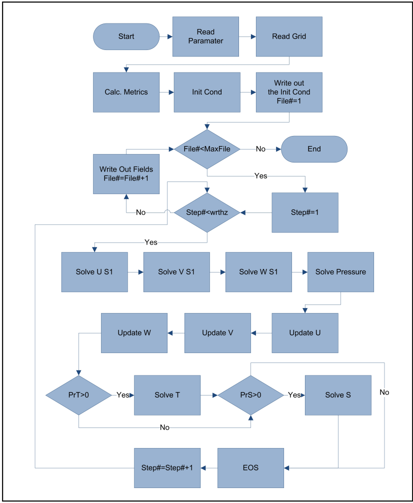
- Waitlist Update: lists are still large for both sections
- Introduction to HPC (Part 1)
- Pretest
  - Required for all/new students
  - Pretest #1: Avg = 7.23/10.0, min=1.0, max=10.0

My serial code is working, why should I parallelize it?

- Compute bound
  - Large number of iterations
  - Long time to converge
- Data bound
  - Memory footprint is larger than RAM (common for science apps)
  - Output may take FOREVER to save/dump/transfer
- contention for resources/other processes
  - What other processes are running on the machine?
  - What other data is running on the network?
  - Timeout problems?

General Curvilinear Environmental Model

- Simulate processes along the coast
- Serial version, Simple Seamount, appx 8k lines
    - 1 km /cell, 100x50x50 (small job)
    - large memory: 100 arrays $\tilde{O}(10^7)$
    - 200k iters ≃ 6 hours simulation time
    - Twall ≃ 70 hours ≃ $O(10^{14})$ ops
- Typical simulations need to run for days/weeks/months/years
    - Jaguar (ORNL), $2x10^5$ cores (≃3GHz, 16GB), $1.75x10^{12}$ flops
    - Could run this job in $0.3x10^{-2}$ seconds (note: est. does not include time to fire up 200k cores)
    - Motivation for parallel computing!



Source: http://www.xsede.org

Porting Code to Parallel has Many Challenges

- I have 200k cores, now what do I do with them?
- How to distribute the data? Is it memory/IO bound)?
- How to distribute the computation? Is it compute bound?
- How do you synchronize the results?

What is Parallel Computing and Why do We Need it?

- Large problems:
  - Biz: BM has 433,000 employees; how do you update paycheck/empoyee data?
  - Science: Hurricane Predictions, GoM is appx 800x800 km $= 6.4\text{x}10^5$ points/array. Typically have 10^2 arrays. Forecast models use multiple models, multiple clusters, take 12 hours (wall clock) to forecast out 3-4 days on parallel computers (probably 500 nodes). See `http://www.srh.noaa.gov/ssd/nwpmodel/html/nhcmodel.htm`, f
- Large data: These models generate massive data sets (Tera-to-peta-bytes per simulation).
  - My coastal ocean model generates Vel(U,V,W), Pressure,Temp,Salinity files once each iteration at 2.5MB/file. 6 hour test run is $2\text{x}10^5$ iterations $= 3\text{x}10^6$ MBytes (but we don't have room so we only store 100 slices).
  - Where do you store the data?
  - How do you move it? Sneaker Net?
- Fast networks are required
- Power consumption is big issue

Typical HPC Computing Resources

- Compute
  - HPC systems & Clusters
  - GPU
  - Clouds
- Archival & Storage
  - high speed servers with large storage
  - Clouds
- Network
- Visualization
- Cyberinfrastructure

HPC Units of Measure

What is the scale of modern parallel resources & computations

Unit	Definition
Flops	floating point operations
Flop/s	floating point operations per second

Prefix	#Flops/sec 2	Bytes	10^n	2^n
Mega	Mflop/s	Mbyte	10^6	$2^{20} = 1{,}048{,}576$
Giga	Gflop/s	Gbyte	10^9	$2^{30} = 1{,}073{,}741{,}824$
Tera	Tflop/s	Tbyte	10^{12}	$2^{40} = 10{,}995{,}211{,}627{,}776$
Peta	Pflop/s	Pbyte	10^{15}	$2^{50} = 1{,}125{,}899{,}906{,}842{,}624$
Exa	Eflop/s	Ebyte	10^{18}	$2^{60} = 1.15292152^{18}$

A Large HPC Machine: Kraken

Hostname	kraken.nics.teragrid.org
Manufacturer	Cray
Model	XT5
Processor Cores	112896
Nodes	9408
Memory	147.00 TB
Peak Performance	1174.00 TFlops
Disk	2400.00 TB

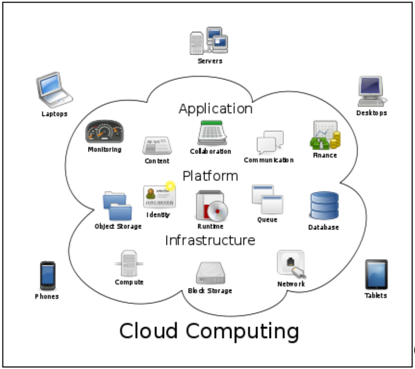
Source: `http://www.xsede.org`

COMP605-Sp15/Images

Cloud Computing

Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). There are many types of public cloud computing:

- Infrastructure as a service (IaaS),
- Platform as a service (PaaS),
- Software as a service (SaaS)
- Storage as a service (STaaS)
- Security as a service (SECaaS)
- Data as a service (DaaS)
- Business process as a service (BPaaS)
- Test environment as a service (TEaaS)
- Desktop as a service (DaaS)
- API as a service (APIaaS)
- Cyberinfrastructure



COMP605 Sp15/Images

Source: http://en.wikipedia.org/wiki/Cloud_computing

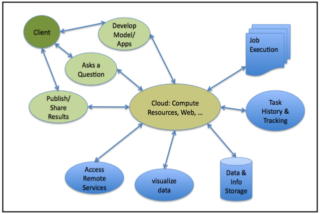
Cyberinfrastructure (CI)

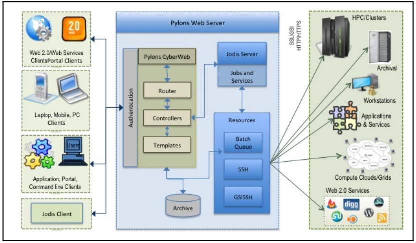
- Cyberinfrastructure - Research environment or infrastructure based upon distributed computer, information and communication technology. (From: Atkins PITCAC Report, 2003)
- Resources: data acquisition, data storage, data management, data integration, data mining, data visualization, computing and information processing services distributed over the Internet beyond the scope of a single institution.*
- Scientific Comptuing: a technological and sociological solution to the problem of efficiently connecting laboratories, data, computers, and people with the goal of enabling derivation of novel scientific theories and knowledge.*
- Cyberinfrastructure = HPC + Cloud/Grid + Networks

* Source: http://en.wikipedia.org/wiki/Cyberinfrastructure

Example of CI Project: CyberWeb

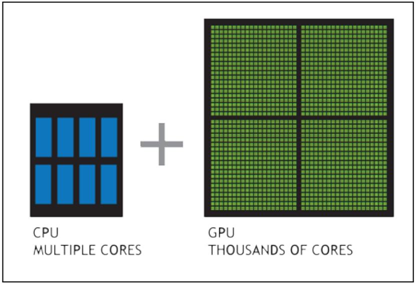
- Provide a bridge between users and high-end resources, emerging technologies and cyberinfrastructure.
- Simplify use by using common/familiar Web and emerging technologies
- Facilitate access to and utilization of Science Applications.
- Apps run in heterogeneous environment
- Plug-n-play mode





GPU Computing

- GPU (graphics processing unit) + CPU accelerates scientific and engineering applications.
- CPUs - a few cores
- GPUs - thousands of smaller, more efficient cores designed for parallel performance.
- Serial code run on the CPU while parallel portions run on the GPU



Source: http://www.nvidia.com

HPC Requirements

- Supercomputers: Moores Law - compute cycles exceeding ability to move data
  - Networks needed to move data around to resources
  - Large data sets (GB in 96, but now peta-bytes)
- Computational apps will always use up resources:
  - Just increase resolution: e.g. $3x3x3 = 27- > 5x5x5 = 125$
- New architectures impacting HPC libraries, programming models, languages
- Security: desire to protect national resources
- NSF (and other gov agencies) spend money to build infrastructure

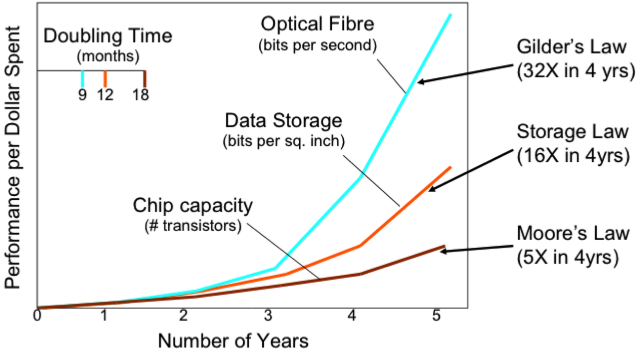
COMP605-Sp15/Images

Technology Growth Laws

- **Moore's Law:** Processing power of a cpu/core doubles every 18 months; corollary, computers become faster and the price of a given level of computing power halves every 18 months (Gordon Moore,70's).
- **Gilder's Law:** Total bandwidth of communication systems triples every twelve months (George Gilder).
- **Metcalfe's Law:** The value of a network is proportional to the square of the number of nodes; so, as a network grows, the value of being connected to it grows exponentially, while the cost per user remains the same or even reduces (Robert Metcalfe, originator of Ethernet)
- **Kryder's Law:** The density of information on hard drives growing faster than Moore's law, and is doubling roughly every 13 months.

Source: `http://www.jimpinto.com/writings/techlaws.html`, and `http://www.scientificamerican.com/article/kryders-law/`

Technology Growth Laws



Source: G. Stix, Triumph of Light. Scientific American. January 2001

HPC Performance

Source: `http://www.top500.org`

HPC and CI evolution

- Explosion of computer technologies in the late 90s had a significant impact on HPC
  - Advances in commodity computers allows researchers to build clusters equivalent to big iron
- Compute TOP 500 numbers:
  - June 2014: National Super Computer Center, Guangzhou
    - Cores: 3,120,000
    - Tflops/s: 54902.4
    - Power: 17808 kW
  - 06 - 367 Tflops/s
    - IBM BlueGene 131072 processors
    - Hardware: 0.7 GHz PowerPC 440
  - 04 - Total 92 TFlops
    - IBM BlueGene (¿360 GF); 32768 processors
    - Hardware: XEON, dual CPU, 2GHz
  - 97 - Total 17 TFlops, Sun NOW (33 GFlops
  - 95 - Total 4.4 TFlops, no clusters

Source: `http://www.top500.org`

Top500

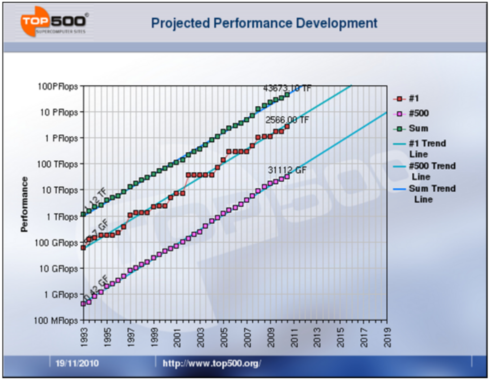


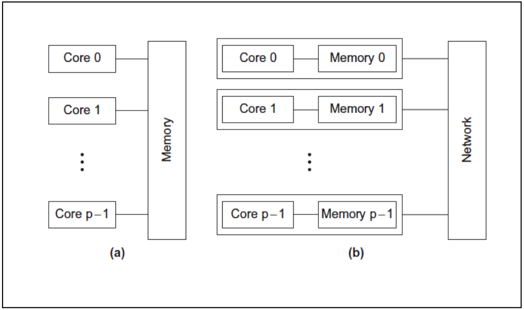
Figure: Source: http://blog.infinibandta.org/tag/petascale/

High Performance Computing - Key Components

- Parallel Systems (clustered, distributed)
- Fast Networks
- Large, fast data storage
- Parallel Software

Types of Parallel Systems

- Shared-memory
  - Cores share computers memory
  - Cores access shared memory locations –¿ must synchronize.
- Distributed-memory
  - Each core has its own, private memory.
  - Cores communicate explicitly by sending messages across a network.



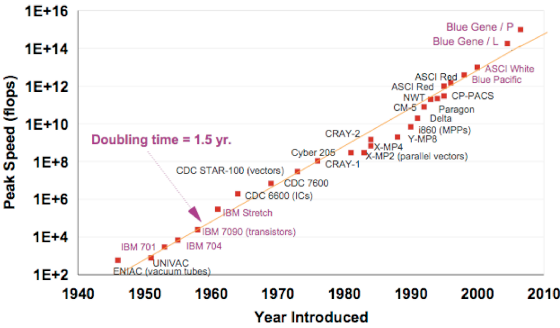
Source: Pacheco, 2011 Textbook

Types of Parallel Computing

- **Concurrent computing:** a program is one in which multiple tasks can be in progress at any instant
  - Stock market transactions, embarrassingly parallel problems.
- **Parallel computing:** a program is one in which multiple tasks cooperate closely to solve a problem
  - CFD, chemical computations, tightly coupled
- **Distributed computing:** a program may need to cooperate with other programs to solve a problem
  - Seti@Home, the cloud, the grid, loosely coupled

HPC Performance Evolution

- J'14 - 54,902 TFLOP/s
  - National Super Computer Center, Guangzhou
  - Cores: 3,120,000
  - Power: 17808 kW
- 06 - 367 TFLOP/s
  - IBM BlueGene
  - 131072 processors
  - Hardware: 0.7 GHz PowerPC 440
- 04 - Total 92 TFLOP/s
  - IBM BlueGene (¿360 GF)
  - 32768 processors
  - Hardware: XEON, dual CPU, 2GHz
- 97 - Sun NOW 33 GFlops
- 95 - Total 4.4 TFlops, no clusters



Source: Left: `http://www.top500.org`
Right: `http://www.research.ibm.com/bluegene/BG_External_Presentation_January_2002.pdf`

HPC Networks: Top500

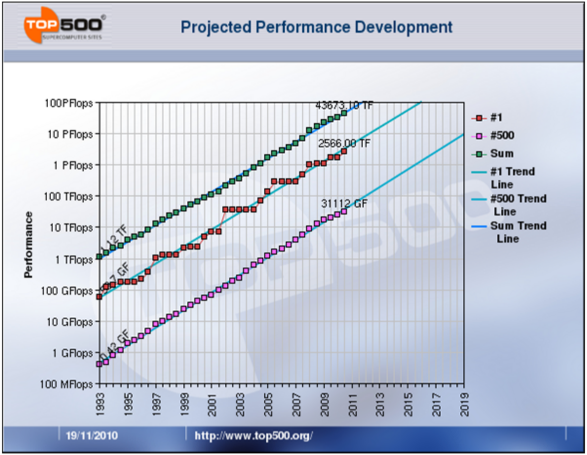
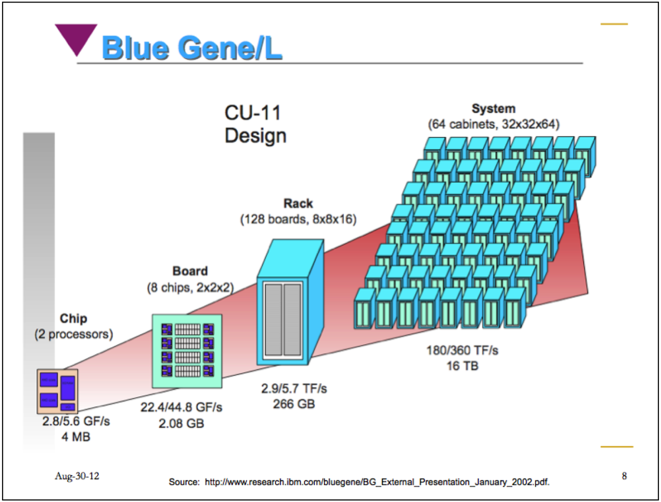


Figure: Source: `http://blog.infinibandta.org/tag/petascale/`

HPC Hardware: Cray Jaguar XT5

- Operational 2005
- Architecture 224,256 AMD Opteron processors
- Speed 1.75 petaflops (peak)
- Cost USD $104M
- Ranking TOP500: 3, June 2011

HPC Hardware: Blue Gene/L Hardware



Source: M. Thomas, CS596, 2007

HPC Hardware: TACC Stampede Cluster

TACC Stampede

Hostname	stampede.tacc.xsede.org
Site	tacc.xsede.org
Organization	Texas Advanced Computing Center
Descriptive Name	TACC Dell PowerEdge C8220 Cluster with Intel Xeon Phi coprocessors (Stampede)
Manufacturer	Dell
Platform	Dell PowerEdge C8220 Cluster with Intel Xeon Phi coprocessors
CPU Type	Intel Xeon E5-2680
Machine Type	Cluster
Operating System	Linux (CentOS)
Contact	XSEDE Help Desk
Processor Cores	102400
Nodes	6400
Memory	200 TB
Peak Performance	9600 TFlops
Disk	14336

Source: `http://www.xsede.org`

- Intel Xeon Dual Processor Micro Architecture
- Intel Xeon Dual Processor Micro Architecture
- Hyper-Threading Technology
- Achieved by duplicating the architectural state on each processor
- Share one set of processor exec resources.
- Architectural state tracks the flow of a program or thread
- Execution resources: add, multiply, load
- Rapid Exec. Engine (REE): exec simultaneously based on instruction queues
- Fetch and Deliver Engine: moves instructions in/out of REE
- Integrated Cache System : Delivers data and instructions to procs; ops at cpu speed
- Reorder and Retire Block: Keeps instructions and program in order, saves state
- System Bus: Increase throughput of apps and BW

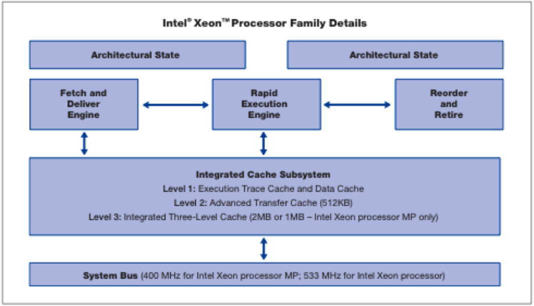
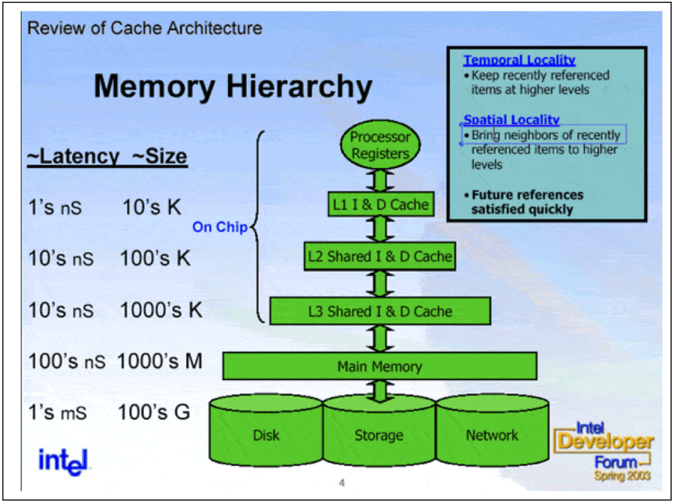


Figure 3: High-level block diagram of the Intel® Xeon™ processor family for servers shows how the various pieces of the microarchitecture relate to each other with Hyper-Threading Technology.

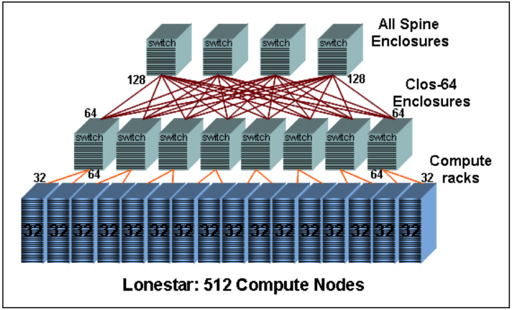
HPC Hardware: Memory



Source: `http://www17.tomshardware.com/cpu/20030811/images/intelgrafik.gif`

HPC Hardware: Interconnection Networks

- Myrinet network for 512 nodes
- 12 switches,each housing up to 16 leaf or spine cards
- each card has 8 fiber ports.
- leaf ports connect to hosts (1750 nodes)
- spine ports are used to form a hierarchical switch fabric.
- groups of 64 nodes are connected into 64 ports on each switch
- 8 ports x 8 leaf cards in 8 switches.

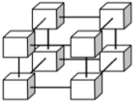


Sources: M. Thomas, CS596, 2007

HPC Hardware - Blue Gene/L Networks

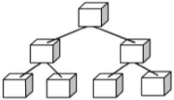
Blue Gene/L - The Networks

- 65536 nodes interconnected with three integrated networks



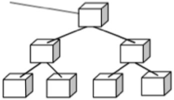
3 Dimensional Torus

- Virtual cut-through hardware routing to maximize efficiency
- 2.8 Gb/s on all 12 node links (total of 4.2 GB/s per node)
- Communication backbone
- 134 TB/s total torus interconnect bandwidth



Global Tree

- One-to-all or all-all broadcast functionality
- Arithmetic operations implemented in tree
- ~1.4 GB/s of bandwidth from any node to all other nodes
- Latency of tree traversal less than 1usec



Ethernet

- Incorporated into every node ASIC
- Disk I/O
- Host control, booting and diagnostics

Aug-30-12 Source: http://www.research.ibm.com/bluegene/BG_External_Presentation_January_2002.pdf. 9

Source: M. Thomas, CS596, 2007

Ethernet Growth

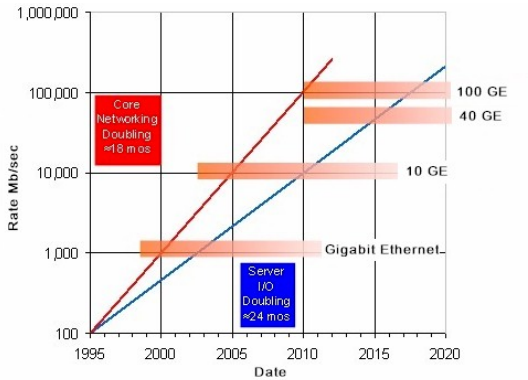


Figure: Source: `http://www.infinibandta.org/content/pages.php?pg=technology_overview`

InfiniBand

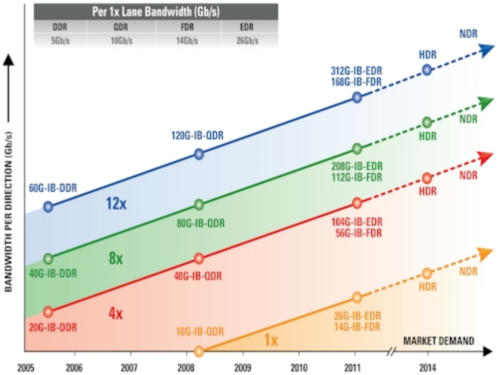


Figure: Source: http://www.infinibandta.org/content/pages.php?pg=technology_overview

Developing Parallel Algorithm Example: Serial Summation

Example

- Compute n values and add them together.
- Serial solution:

```
sum = 0;
for (i = 0; i < n; i++) {
   x = Compute_next_value(. . .);
   sum += x;
}
```

Simple Parallel Sum: Distribute Work to Different Cores

Example (cont.)

- We have p cores, p much smaller than n.
- Each core performs a partial sum of approximately n/p values.

```
my_sum = 0;
my_first_i = . . . ;
my_last_i = . . . ;
for (my_i = my_first_i; my_i < my_last_i; my_i++) {
   my_x = Compute_next_value( . . .);
   my_sum += my_x;
}
```

Each core uses it's own private variables and executes this block of code independently of the other cores.

Simple Parallel Sum: Each Core Holds Its Results

- After each core completes execution of the code, is a private variable my_sum contains the sum of the values computed by its calls to Compute_next_value.
- Ex 8 cores, n = 24, then the calls to Compute_next_value return:
- [1,4,3] [9,2,8] [5,1,1] [5,2,7] [2,5,0] [4,1,8] [6,5,1] [2,3,9]
- Once all the cores are done computing their private my_sum, they form a global sum by sending results to a designated master core which adds the final result.

Source: Pacheco, 2011 Textbook

Simple Parallel Sum: Gather Results to One Core

Example (cont.)

```
if (I'm the master core) {
   sum = my_x;
   for each core other than myself {
      receive value from core;
      sum += value;
   }
} else {
   send my_x to the master;
}
```

Simple Parallel Sum: Data Distribution

Example (cont.)

Core	0	1	2	3	4	5	6	7
my_sum	8	19	7	15	7	13	12	14

Global sum

$$8 + 19 + 7 + 15 + 7 + 13 + 12 + 14 = 95$$

Core	0	1	2	3	4	5	6	7
my_sum	95	19	7	15	7	13	12	14

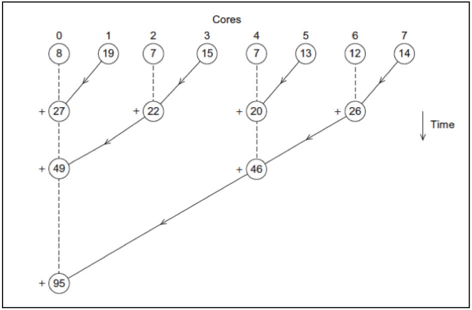
Better parallel algorithm: Reduce Master Core Workload

- After core completes execution, local variable my_sum contains the sum of the values computed by its calls to Compute_next_value.
    - Pair the cores so that core 0 adds its result with core 1s result.
    - Core 2 adds its result with core 3s result, etc.
    - Work with odd and even numbered pairs of cores.
    - Repeat the process now with only the evenly ranked cores.
    - Core 0 adds result from core 2.
    - Core 4 adds the result from core 6, etc.
- Now cores divisible by 4 repeat the process, and so forth, until core 0 has the final result.

Source: Pacheco, 2011 Textbook

Simple Parallel Sum: Tree-structured Addition

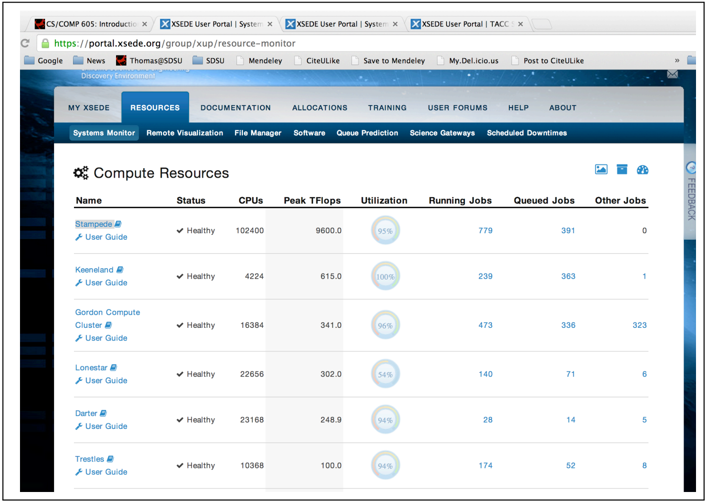
- Soltn 1: master core - 7 recvs & 7 adds.
- Soltn 1: master core - 3 recvs & 3 adds.
- Improvement greater than 2!
- for 10^3 cores,
  - Soltn 1: 999 recvs & 999 adds.
  - Soltn 1: 10 recvs & 10 adds.



Ways to Improve Parallel Algorithms

- Develop parallel version of serial problems
  - task-paralellism: distribute different tasks among cores (stock market transactions)
  - data-paralellism: distribute the data among cores, each core does the same computation
- Develop parallel hardware
  - fast core speed, large Ram memory, fast interconnections
  - develop parallel software libraries to simplify programming env.

HPC Hardware - NSF XSEDE Project



Name	Status	CPUs	Peak TFlops	Utilization	Running Jobs	Queued Jobs	Other Jobs
Stampede User Guide	✔ Healthy	102400	9600.0	95%	779	391	0
Keeneland User Guide	✔ Healthy	4224	615.0	100%	239	363	1
Gordon Compute Cluster User Guide	✔ Healthy	16384	341.0	96%	473	336	323
Lonestar User Guide	✔ Healthy	22656	302.0	54%	140	71	6
Darter User Guide	✔ Healthy	23168	248.9	94%	28	14	5
Trestles User Guide	✔ Healthy	10368	100.0	94%	174	52	8

Source: http://www.xsede.org

Next Time

- Next class: 01/24/17
- Topics: Unix basics/Using student cluster, etc., Introduction to HPC (Part 2)
- Homework 1 will be posted today: Due date: 02/02/16 at beginning of class.