CS395T: Introduction to Scientific and Technical Computing

Instructors:

Dr. Karl W. Schulz, Research Associate, TACC
Dr. Victor Eijkhout, Research Scientist, TACC

Outline

- Finish Architecture Discussion
- Ranger Intro - *pictures O' the week*
- Class Accounts
  - *Do's and Don'ts*
- Batch System Introduction and Usage

Data Reuse

- Performance is limited by data transfer rate
- High performance if data items are used multiple times
- Example: vector addition $x_i=x_i+y_i$: 1op, 3 mem accesses
- Example: inner product $s=s+x_i*y_i$: 2op, 2 mem access (s in register; also no writes)

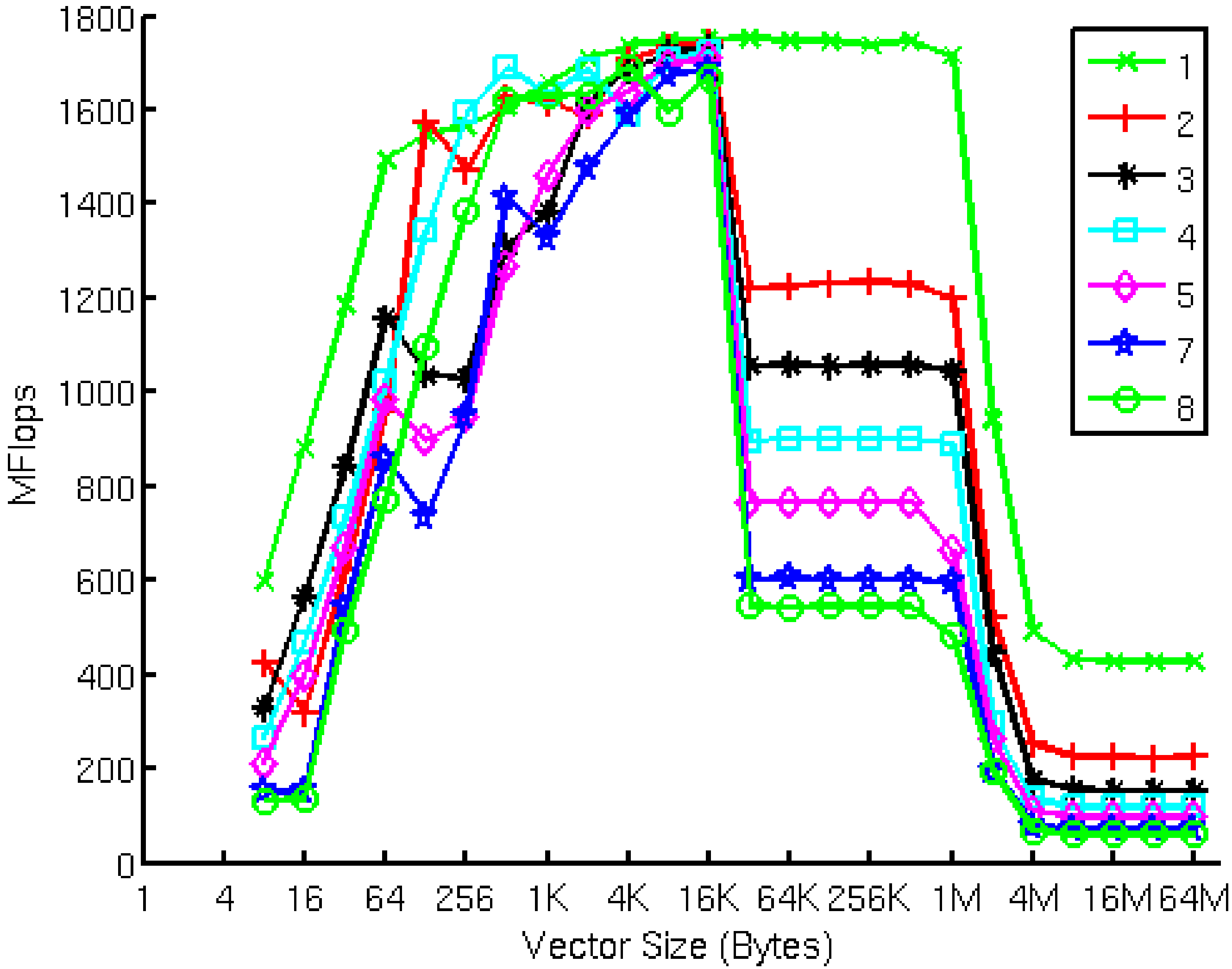
Programming Strategies: Contiguous Access

- Avoid strides: cache lines contain 4 (or so) words, might as well use them all
- Example: dot product, sequential access of the vectors
- Strided dot product:

```
sum=0.;
for (j=0; j < stride; ++j)
  for(i=j; i < n; i+=stride)
    sum += a[i]*b[i];
```

Not all elements on a cache line used.

Dot Product Performance

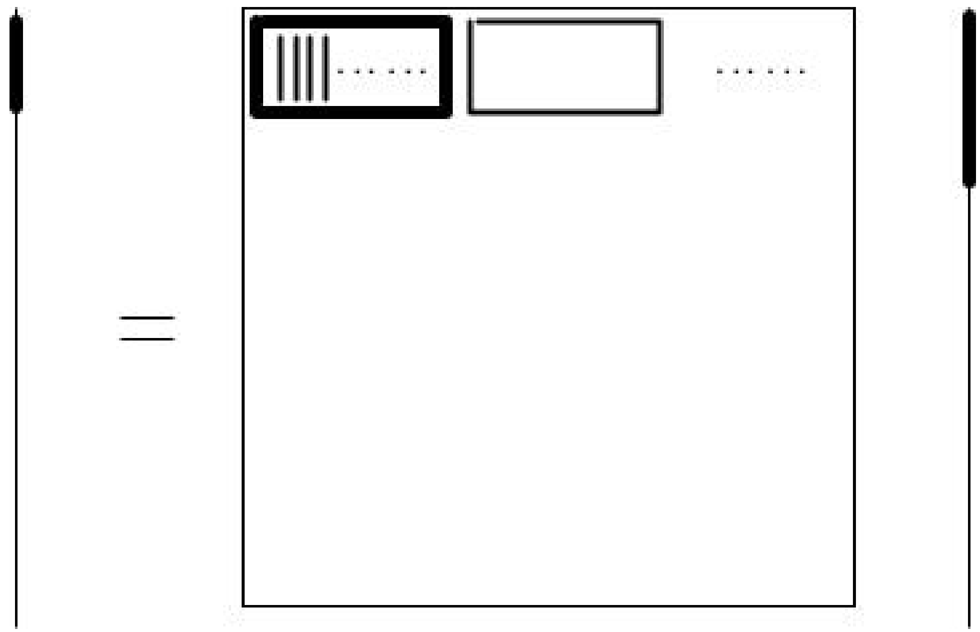


What's Going On?

- Small vectors are noisy
  - probably not enough work to be measuring well
  - even still non-stride-1 access foil the plans of the hardware prefetcher
- Eventually everyone gets to the peak
  - ~1.8 GFlops = ~20% of peak
  - not far from what we predicted
  - probably some improvement yet to be had
- For *stride != 1* we see the L1 (32K) cache size boundary
- For *stride == 1* prefetching and other latency hiding tricks let the processor maintain performance
- Everybody hits the L2 (4MB) cache size boundary pretty hard

Programming Strategies: Blocked Algorithms

- Long vectors are flushed from cache: break up in smaller blocks
- Reuse of input vector (limited)
- Use of cache lines in matrix



More on blocked algorithms

- This gets tricky fast
- Matrix-matrix multiply is triple loop; blocked is 6-deep loop
- Choice of blocking sizes is complicated
- Loop exchange to aim for L1 or L2 reuse
  (Atlas vs Goto approach)

➢ *The Solution*: Use optimized math libraries whenever possible to do routine linear algebra in your applications (more on libraries coming later in the class)

Pipelining

Pipeline

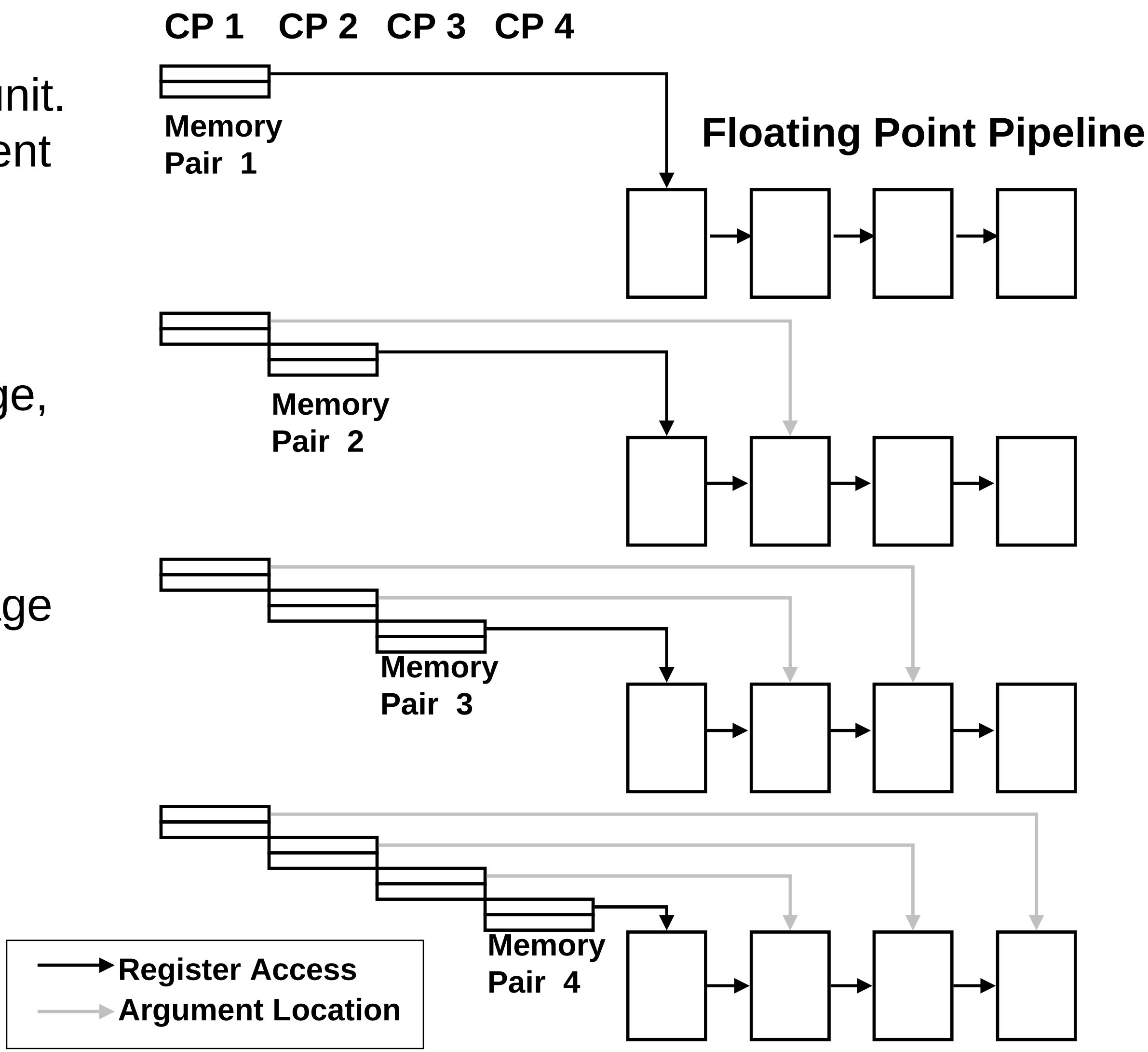
A serial multistage functional unit. Each stage can work on different sets of independent operands simultaneously.

After execution in the final stage, first result is available.

Latency = # of stages * CP/stage

CP/stage is the same for each stage and usually 1.

4-Stage FP Pipe



Branch Prediction

- The “instruction pipeline” is all of the processing steps (also called segments) that an instruction must pass through to be “executed”.
- Higher frequency machines have a larger number of segments.
- Branches are points in the instruction stream where the execution may jump to another location, instead of executing the next instruction.
- For repeated branch points (within loops), instead of waiting for the loop to branch route outcome, it is predicted.

Pentium III processor pipeline

1	2	3	4	5	6	7	8	9	10

Pentium 4 processor pipeline

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

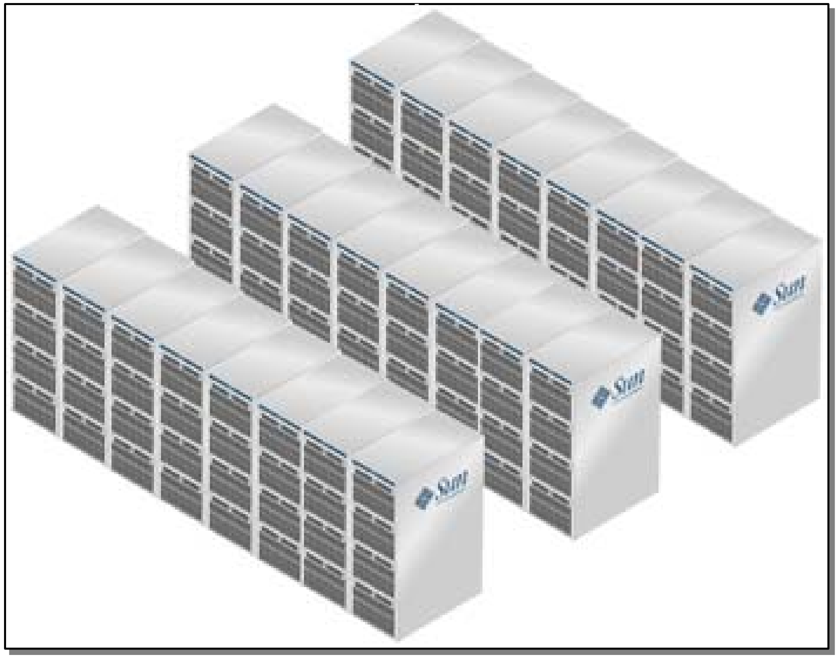
Misprediction is more “expensive” on Pentium 4’s.

TACC's Ranger System

First NSF Track2 System: 500+ Tflops!

- TACC selected for first NSF ‘Track2’ HPC system
  - $30M system acquisition
  - Sun is the vendor
  - Competed against almost every open science HPC center

- TACC, ICES, Cornell, ASU operating/supporting system four 4 years ($29M)



Ranger System Configuration

- Compute Power - 504 Teraflops Peak Performance
  - 3,936 Sun four-socket, quad-core compute nodes
  - 15,744 AMD Opteron "Barcelona" processors
    - Quad-core, four flops/cycle (dual pipelines)
- Memory
  - 2 GB/core, 32 GB/node, 125 TB total
  - 132 GB/s aggregate bandwidth
- Infiniband interconnect
  - Full non-blocking 7-stage Clos fabric
  - Low latency (~2.3 μsec), high-bandwidth (~950 MB/s)

Impact in NSF TeraGrid

- 460M CPU hours to TeraGrid per year
  - more than double current total capacity of all TG HPC systems
  - 1.8 Billion CPU hours over operational life
- 529 Teraflops peak
  - 2x total performance of all TG HPC systems
  - 8x top TG HPC system in performance, memory, disk
- Balanced, general-purpose capability system
  - More than 60,000 cores available
  - Unprecedented scaling opportunities for computational science and research
- Re-establish NSF as a leader in HPC
- *Jumpstarts progress to petascale for entire US academic research community*

Ranger User Environment

- The overall look and feel of *Ranger* from the user perspective will be very similar to our current Linux cluster
  - Full Linux OS w/ hardware counter patches on login and compute nodes (2.6.12.6 is starting working kernel)
  - Lustre File System
    - $HOME, and multiple $WORKS will be available
    - Largest $WORK will be ~1PB total
  - Standard 3rd party packages
  - Infiniband using next generation of Open Fabrics
  - MVAPICH and OpenMPI (MPI1 and MPI2)

- Suite of compilers
  - Portland Group PGI
  - Sun Studio
  - PathScale
  - *Possibly the Intel compiler*

Ranger Disk Subsystem - *Lustre*

- Disk system (OSS) is based on Sun x4500 “Thumper” servers - similar to TiTech installation
  - Each server has 48 SATA II 500 GB drives (24TB total) - running internal software RAID
  - Dual Socket/Dual-Core Opterons @ 2.6 GHz
  - Downside is that these nodes have PCI-X - raw I/O bandwidth can exceed a single PCI-X 4X Infiniband HCA
  - 72 Servers Total: 1.7 PB raw storage
- Metadata Servers (MDS) based on Sun Fire x4600s
- MDS is Fibre-channel connected to 9TB Flexline Storage
- Target Performance
  - Aggregate bandwidth: 40 GB/sec

Design:
- Top loading Disks
- Front to rear airflow
- Redundant fans
- Passive Backplane
- No wires in box

Reliability/Availability
- Enterprise class SATA disks
- 1M hours MTBF
- RAID 0, 1, 5, 10
- Redundant Power
- Hot-swap FRUs

Ranger System Configuration

Logical Volume Name	Estimated Raw Capacity	Target Usage
WORK1	~850 PB	Large temporary storage; not backed up, purged periodically
WORK2	~250 TB	Large allocated storage; not backed up, quota enforced
PROJECTS	2 TB	Repository for TeraGrid Community Software
HOME1	2 TB	Permanent user storage; automatically backed up, quota enforced
HOME2	2 TB	Permanent user storage; automatically backed up, quota enforced
HOME3	2 TB	Permanent user storage; automatically backed up, quota enforced

Ranger Space, Power and Cooling

- System Power: 3.0 MW total
- System: 2.4 MW
  - ~90 racks, in 6 row arrangement
  - ~100 in-row cooling units
  - ~4000 ft^2 total footprint
- Cooling: ~0.6 MW
  - In-row units fed by three 400-ton chillers
  - Enclosed hot-aisles
  - Supplemental 280-tons of cooling from CRAC units
- Observations:
  - Space less an issue than power
  - Cooling > 25kW per rack difficult
  - Power distribution a challenge, more than 1200 circuits

Ranger Project Timeline

Sep06	award, press, relief, beers
1Q07	equipment begins arriving
2Q07	facilities upgrades complete
3Q07	very friendly users
4Q07	more early users
Dec07	production, many beers
Jan08	allocations begin

Note: all US academics are eligible to apply for a TeraGrid/Ranger allocation:

https://pops-submit.ci-partnership.org/

Ranger: External Infrastructure

Ranger: PDUs and In-Row Coolers

- APC In-row coolers are installed and plumbed
- Compute Racks will slide in between the coolers which are heat exchangers drawing from the hot aisles, exhausting ambient into the cold aisles

TACC

Ranger: Machine Room Layout

Computer Accounts

Computer Accounts

- Lonestar
  - Production Resource for UT and the NSF TeraGrid Community
  - 1460 Dell 1955 nodes
  - 2 dual-core 2.6GHz Intel Xeon (Woodcrest) Processors/node
  - 8 GB RAM/node
  - Infiniband Interconnect
  - Currently listed as #15 on the Top500 List

“Production”

- Jobs run in a managed environment
  - login to the login node
  - submit jobs to the scheduler
  - wait
  - collect results
- Running programs on the login node highly discouraged
  - avoid resource intensive tasks
  - exceptions include compilers, “standard” UNIX commands (ls, mkdir, cp, mv, etc.)

Lonestar Login

- SSH only
- UNIX users

  ssh username@lonestar.tacc.utexas.edu
- Windows users: Get a client
  - PuTTY
    - http://www.chiark.greenend.org.uk/~sgtatham/putty/
    - PuTTY can be gotten from Bevoware as well
      - https://www.utexas.edu/its/bevoware/download/
  - or use Cygwin and follow the UNIX instructions
    - http://www.cygwin.com/

TACC

First Login

- Login using your username and password
  - both are **case sensitive**
- Change your password

```
lslogin1% passwd
Changing password for user istc00.
Changing password for istc00
(current) UNIX password:
New UNIX password:
Retype new UNIX password:
lslogin1%
```

- Logout

```
lslogin1% logout
```

TACC

Lonestar Usage

- If you don’t know what you’re doing, **WAIT!**
- If you’re already running on Lonestar, feel free to use either your account or your class account for assignments
- **DON’T** use the class account to do your research
  - your advisor needs to setup a project and add you to it for research usage

Login Problems

- Send me email
  - karl@tacc.utexas.edu
  - eijkhout@tacc.utexas.edu

- Subject
  - CS395T Can’t login to Lonestar
- Include any error message that you saw

Batch Systems

Batch Systems

- In a number of scientific computing environments, multiple users must *share* a compute resource:
  - research clusters
  - supercomputing centers

- On multi-user HPC clusters, the *batch system* is a key component for aggregating compute nodes into a single, sharable computing resource

- The batch system becomes the “nerve center” for coordinating the use of resources and controlling the state of the system in a way that must be “fair” to its users

- As current and future *expert* users of large-scale compute resources, you need to be familiar with the basics of a batch system

Batch Systems

- The core functionality of all batch systems are essentially the same, regardless of the size or specific configuration of the compute hardware:
  - Multiple Job Queues:
    - queues provide an orderly environment for managing a large number of jobs
    - queues are defined with a variety of limits for maximum run times, memory usage, and processor counts; they are often assigned different priority levels as well
    - may be interactive or non-interactive
  - Job Control:
    - submission of individual jobs to do some work (eg. serial, or parallel HPC applications)
    - simple monitoring and manipulation of individual jobs, and collection of resource usage statistics (e.g., memory usage, CPU usage, and elapsed wall-clock time per job)
  - Job Scheduling
    - policy which decides priority between individual user jobs
    - allocates resources to scheduled jobs

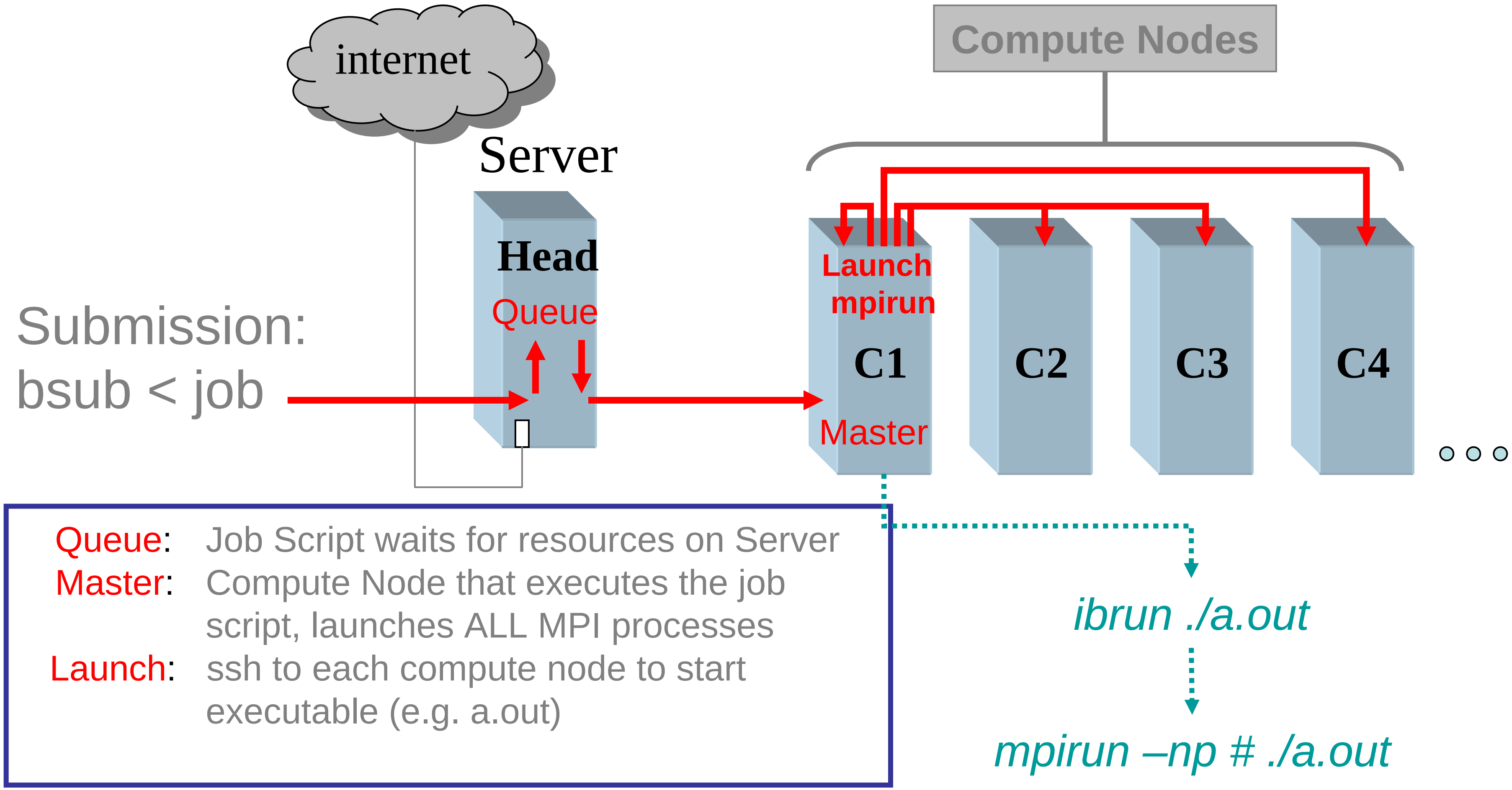
Batch Systems

- Job Scheduling Policies:
  - the scheduler must decide how to prioritize all the jobs on the system and allocate necessary resources for each job (processors, memory, file-systems, etc)
  - scheduling process can be easy or non-trivial depending on the size and desired functionality
    - ***first in, first out (FIFO)*** scheduling: jobs are simply scheduled in the order in which they are submitted
    - ***political*** scheduling: enables some users to have more priority than others
    - ***fairshare*** scheduling, scheduler ensures users have equal access over time
  - Additional features may also impact scheduling order:
    - **advanced reservations** - resources can be reserved in advance for a particular user or job
    - **backfill** - can be combined with any of the scheduling paradigms to allow smaller jobs to run while waiting for enough resources to become available for larger jobs
      - back-fill of smaller jobs helps maximize the overall resource utilization
      - back-fill can be your friend for small duration jobs

Batch Systems

- Common batch systems you may encounter in scientific computing:
  - Platform LSF
  - PBS
  - Loadleveler (IBM)
  - SGE
- All have similar functionality but different syntax
- Reasonably straight forward to convert your job scripts from one system to another
- Above all include specific batch system directives which can be placed in a shell script to request certain resources (processors, queues, etc).
- We will focus on LSF primarily since it is the system running on Lonestar

Batch Submission Process



TACC

LSF Batch System

- *Lonestar* uses Platform LSF for both the batch queuing system and scheduling mechanism (provides similar functionality to PBS, but requires different commands for job submission and monitoring)
- LSF includes *global fairshare*, a mechanism for ensuring no one user monopolizes the computing resources
- Batch jobs are submitted on the front end and are subsequently executed on compute nodes as resources become available
- Order of job execution depends on a variety of parameters:
  - Submission Time
  - Queue Priority: some queues have higher priorities than others
  - Backfill Opportunities: small jobs may be back-filled while waiting for bigger jobs to complete
  - Fairshare Priority: users who have recently used a lot of compute resources will have a lower priority than those who are submitting new jobs
  - Advanced Reservations: jobs my be blocked in order to accommodate advanced reservations (for example, during maintenance windows)
  - Number of Actively Scheduled Jobs: there are limits on the maximum number of concurrent processors used by each user

Lonestar Queue Definitions

Queue Name	Max Runtime	Min/Max Procs	SU Charge Rate	Use
normal	24 hours	2/512	1.0	Normal usage
high	24 hours	2/512	1.8	Higher priority usage
development	30 min	1/32	1.0	Debugging and development Allows *interactive* jobs
hero	24 hours	>512	1.0	Large job submission Requires special permission
serial	12 hours	1/1	1.0	For serial jobs. No more than 4 jobs/user
request				Special Requests
spruce				Debugging & development, special priority, urgent comp. env.
systest				System Use (*TACC Staff only*)

TACC

LSF Fairshare

- A global fairshare mechanism is implemented on Lonestar to provide fair access to its substantial compute resources
- Fairshare computes a dynamic priority for each user and uses this priority in making scheduling decisions
- Dynamic priority is based on the following criteria
  - Number of shares assigned
  - Resources used by jobs belonging to the user:
    - Number of job slots reserved
    - Run time of running jobs
    - Cumulative actual CPU time (not normalized), adjusted so that recently used CPU time is weighted more heavily than CPU time used in the distant past

LSF Fairshare

- **`bhpart`**: Command to see current fairshare priority. For example:

```
lslogin1--> bhpart -r
HOST_PARTITION_NAME:  GlobalPartition
HOSTS:  all

SHARE_INFO_FOR: GlobalPartition/
USER/GROUP   SHARES  PRIORITY  STARTED  RESERVED  CPU_TIME  RUN_TIME
avijit          1       0.333      0         0         0.0         0
chona           1       0.333      0         0         0.0         0
ewalker         1       0.333      0         0         0.0         0
minyard         1       0.333      0         0         0.0         0
phaa406         1       0.333      0         0         0.0         0
bbarth          1       0.333      0         0         0.0         0
milfeld         1       0.333      0         0         2.9         0
karl            1       0.077      0         0     51203.4         0
vmcalo          1       0.000    320         0   2816754.8   7194752
```

Priority

Commonly Used LSF Commands

bhosts	Displays configured compute nodes and their static and dynamic resources (including job slot limits)
lsload	Displays dynamic load information for compute nodes (avg CPU usage, memory usage, available /tmp space)
bsub	submits a batch job to LSF
bqueues	displays information about available queues
bjobs	displays information about running and queued jobs
bhist	displays historical information about jobs
bstop	suspends unfinished jobs
bresume	resumes one or more suspended jobs
bkill	Sends signal to kill, suspend, or resume unfinished jobs
bhpart	Displays global fairshare priority
lshosts	Displays hosts and their static resource configuration
lsuser	Shows user job information

Note: most of these commands support a "-l" argument for long listings. For example: `bhist –l <jobID>` will give a detailed history of a specific job. Consult the man pages for each of these commands for more information.

TACC

LSF Batch System

- LSF Defined Environment Variables:

LSB_ERRORFILE	name of the error file
LSB_JOBID	batch job id
LS_JOBPID	process id of the job
LSB_HOSTS	list of hosts assigned to the job. Multi-cpu hosts will appear more than once (may get truncated)
LSB_QUEUE	batch queue to which job was submitted
LSB_JOBNAME	name user assigned to the job
LS_SUBCWD	directory of submission, i.e. this variable is set equal to $cwd when the job is submitted
LSB_INTERACTIVE	set to ‘y’ when the –I option is used with bsub

LSF Batch System

- Comparison of LSF, PBS and Loadleveler commands that provide similar functionality

LSF	PBS	Loadleveler
bresume	qrls | qsit	llhold -r
bsub	qsub	llsubmit
bqueues	qstat	llclass
bjobs	qstat	llq
bstop	qhold	llhold
bkill	qdel	llcancel

Batch System Concerns

- Submission (need to know)
  - Required Resources
  - Run-time Environment
  - Directory of Submission
  - Directory of Execution
  - Files for stdout/stderr Return
  - Email Notification
- Job Monitoring
- Job Deletion
  - Queued Jobs
  - Running Jobs

LSF: Basic MPI Job Script

```
#!/bin/csh
#BSUB -n 32
#BSUB -J hello
#BSUB -o %J.out
#BSUB -e %J.err
#BSUB -q normal
#BSUB -P A-ccsc
#BSUB -W 0:15

echo "Master Host = "`hostname`
echo "LSF_SUBMIT_DIR: $LS_SUBCWD"
echo "PWD_DIR: "`pwd`

ibrun ./hello
```

- Total number of processes
- Job name
- Stdout Output file name (%J = jobID)
- Stderr Output file name
- Submission queue
- Your Project Name
- Max Run Time (15 minutes)

Echo pertinent environment info

Execution command

Parallel application manager and mpirun wrapper script

executable

LSF: Extended MPI Job Script

```
#!/bin/csh
#BSUB -n 32
#BSUB -J hello
#BSUB -o %J.out
#BSUB -e %J.err
#BSUB -q normal
#BSUB -P A-ccsc
#BSUB -W 0:15
#BSUB -w 'ended(1123)'
#BSUB -u karl@tacc.utexas.edu
#BSUB -B
#BSUB -N

echo "Master Host = "`hostname`
echo "LSF_SUBMIT_DIR: $LS_SUBCWD"

ibrun ./hello
```

Total number of processes

Job name

Stdout Output file name (%J = jobID)

Stderr Output file name

Submission queue

Your Project Name

Max Run Time (15 minutes)

Dependency on Job <1123>

Email address

Email when job begins execution

Email job report information upon completion

TACC