Introduction to Supercomputing

Chris Bording, Daniel Grimwood,

Rebecca Hartman-Baker, Nicola Varini

iVEC

Course Objectives

- Know how supercomputers fit into your workflow.
- Understand how iVEC supercomputers are shared.
    - Merit allocation.
    - Queuing systems.
- Understand which iVEC supercomputer to use.
    - Supercomputer hardware.
    - Scalable and parallel algorithms.
    - Match the supercomputer to the problem.
    - Realistic performance expectations.
- Know how to run jobs on iVEC supercomputers.
    - Use the queuing system.
    - Modules.
    - Stage data.

I. WORKFLOWS

eResearch Workflows

The primary driver of supercomputers is to do good research. They are part of a bigger workflow.

Research enabled by computers is called eResearch.

eResearch examples:

- Simulations (e.g. prediction, testing theories)
- Data analysis
- Visualisation (e.g. interpretation, quality assurance)
- Combining resources (e.g. data mashups, data mining)

Common Research Workflow

Measurement

→

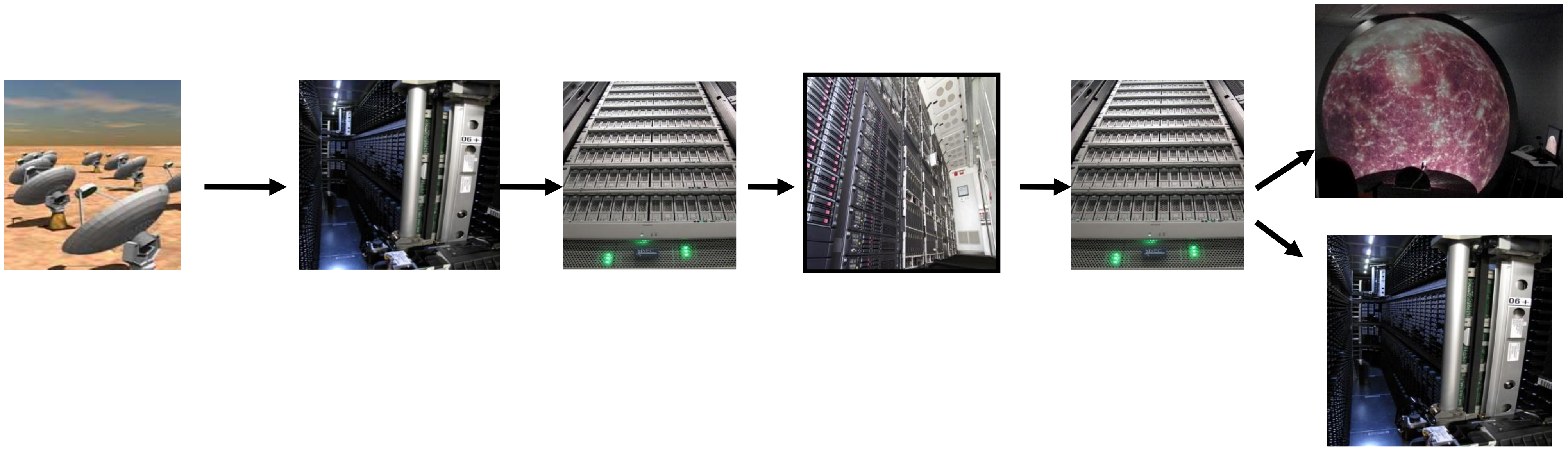
Black box
(processing)

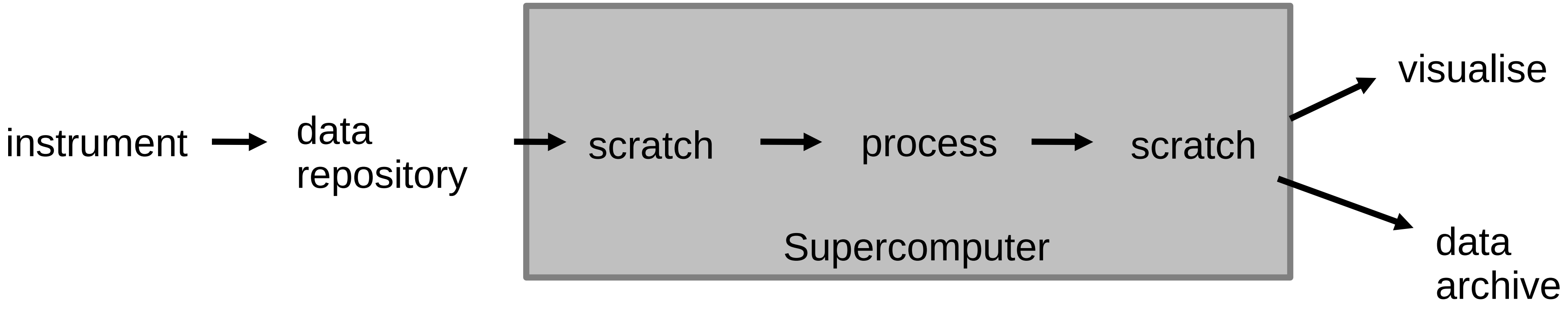
→

Interpretation

Outcomes:
- New knowledge
- Model Validation

eResearch Workflow





Workflow Example

- Think of the everyday example of baking a cake.
- It is a sequence of tasks that follow a recipe.
- Some tasks are independent
  - Can be done in any order.
  - Can be done in parallel.
- Some have prerequisites.
  - Must be done sequentially.

Workflow Concepts

Staging – moving data to where you need it before you need it, or moving it away after you've finished.

Prerequisites – what you need to do something.

Dependencies – relies on something else, usually a prerequisites

Serial Processing – doing tasks one after the other.

Parallel Processing – doing tasks at the same time.

Workflow Concepts

Coarse-grained parallelism – parallelising high-level tasks.

Fine-grained parallelism – parallelising at a low level.

Synchronous – carry out tasks in a coordinated way.

Asynchronous – carry out tasks independently.

Baking a Cake - Staging

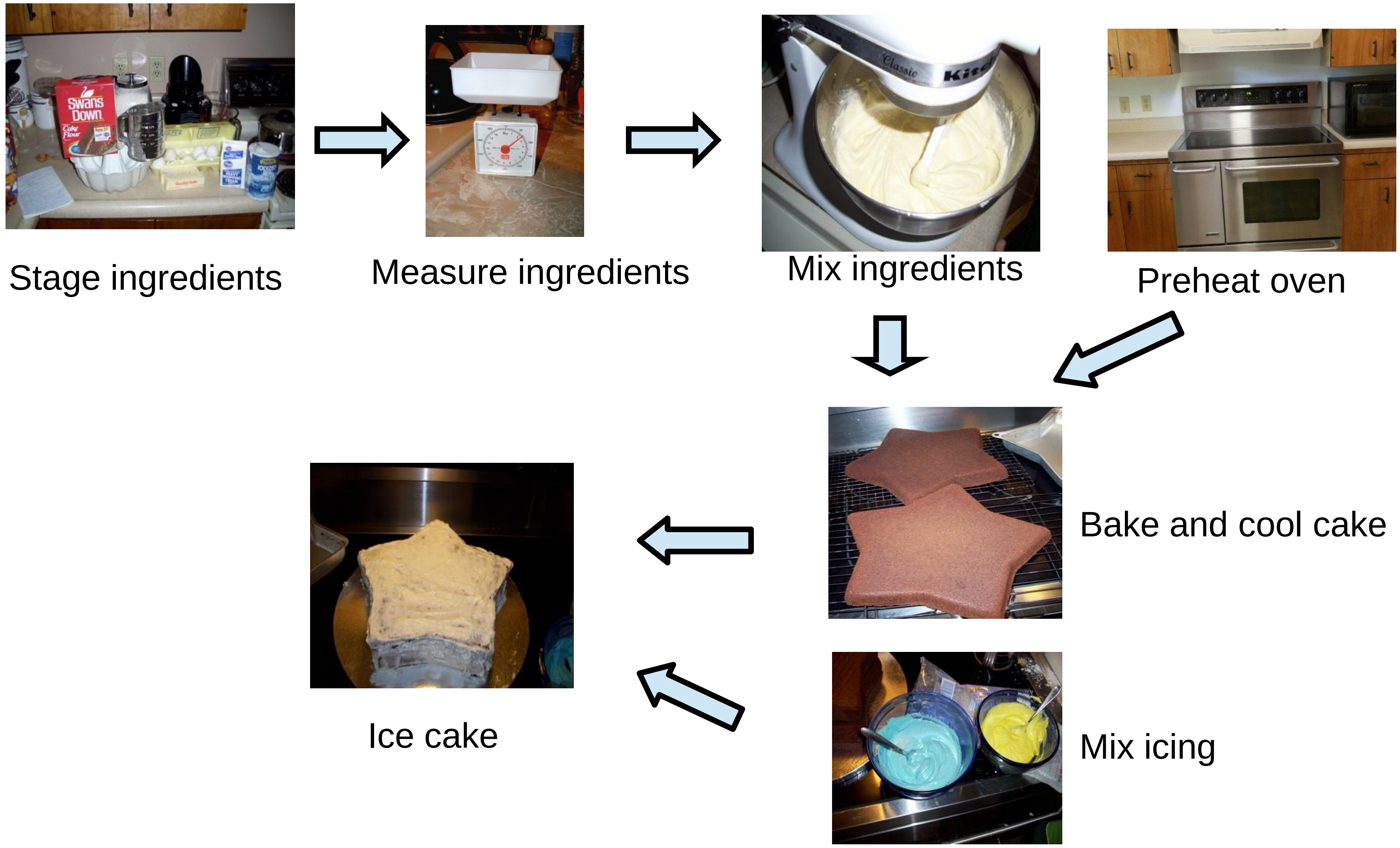
- "Staging" the ingredients improves access time.
- Each ingredient does not depend on others, so can be moved in parallel, or at different times.

Supermarket

Kitchen pantry

Kitchen bench

Baking a Cake - Process



Baking a Cake - Finished

Either:

- Sample it for quality.
- Put it in a shop for others to browse and buy.
- Store it for later.
- Eat it!

- Clean up!

Review / Discussion

In the cake baking workflow, arrows show dependencies.

- which steps can be broken up and parallelised?
- which steps can be done in parallel?
- what is the optimal number of people working on it to get the job done quicker? One, infinity, or somewhere in between?
- what is the best way to get 100 cakes?

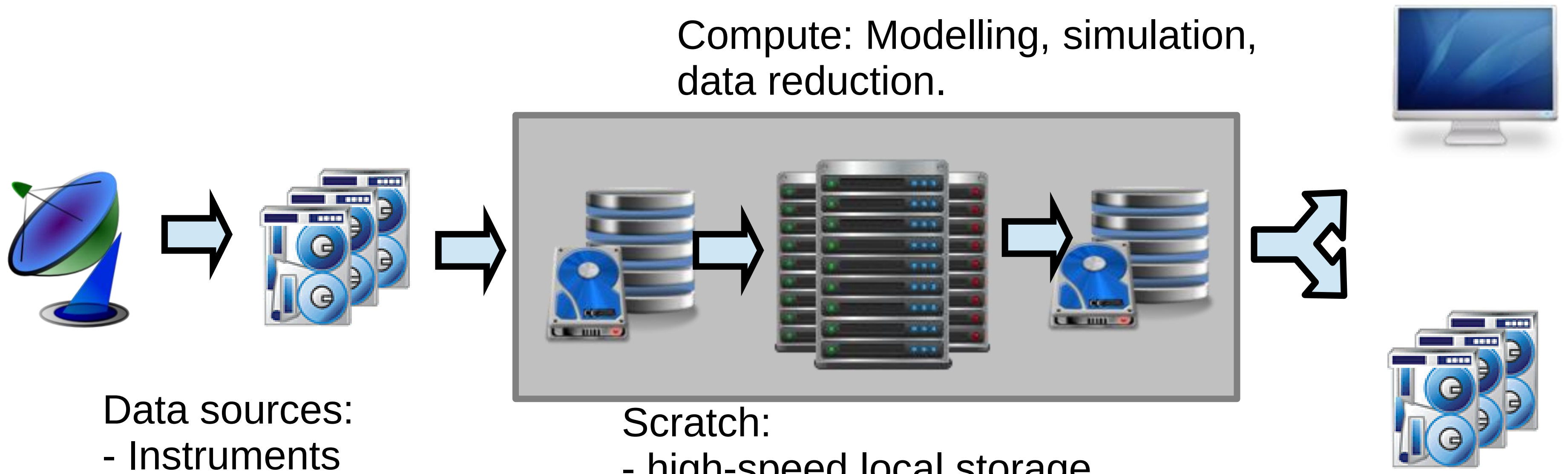
Conclusion from the Kitchen

- If you need many cakes, get many people to bake them independently in their own kitchens. Alternatively, turn it into a production line so everyone is busy most of the time.
  - What is your computational research goal – high throughput of many jobs, or to do one job as fast as possible?
- If you need one cake, experience as well as trial and error will tell you the optimal number of people to help.
- There are different ways to break up the workload.
  - Often high level parallelism scales better, whereas low level parallelism gets small tasks done quicker.

Typical iVEC Workflow

Visualise:
- To understand
- Quality control

Compute: Modelling, simulation, data reduction.

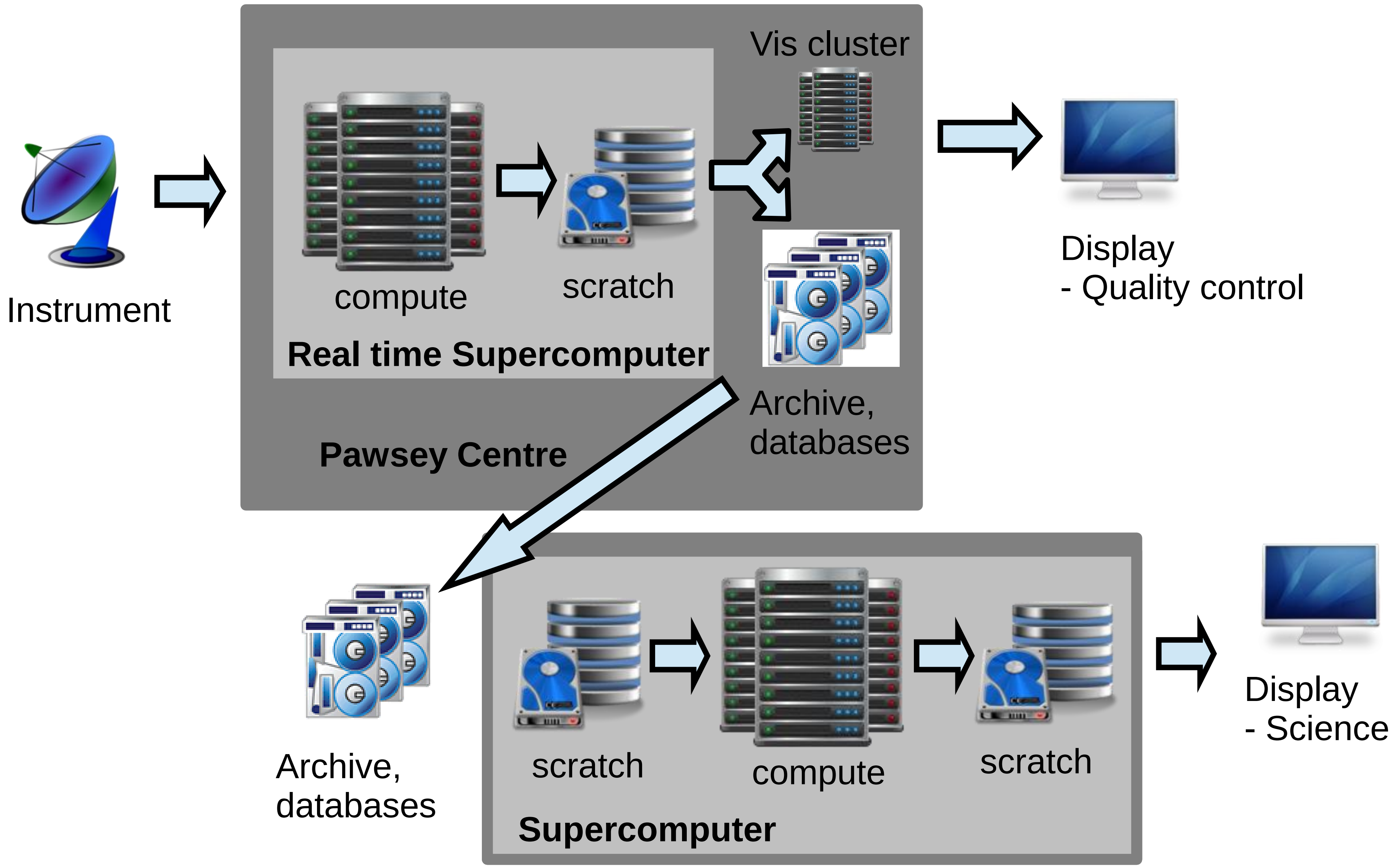


Data sources:
- Instruments
- Databases
- Repositories
- Input files

Scratch:
- high-speed local storage, to keep the compute busy. Stage data to this.

Archive:
- Input for somebody else
- A record of results

ASKAP

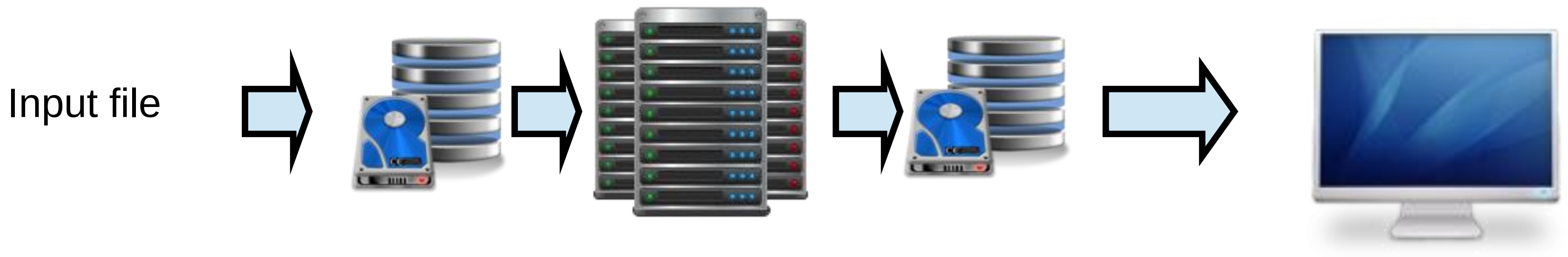


Instrument
Vis cluster
compute
scratch
Real time Supercomputer
Archive, databases
Pawsey Centre
Display - Quality control
Archive, databases
scratch
compute
scratch
Supercomputer
Display - Science

Ab Initio Calculations

Small input, lots of compute

- Quantum Chemistry
- Protein folding



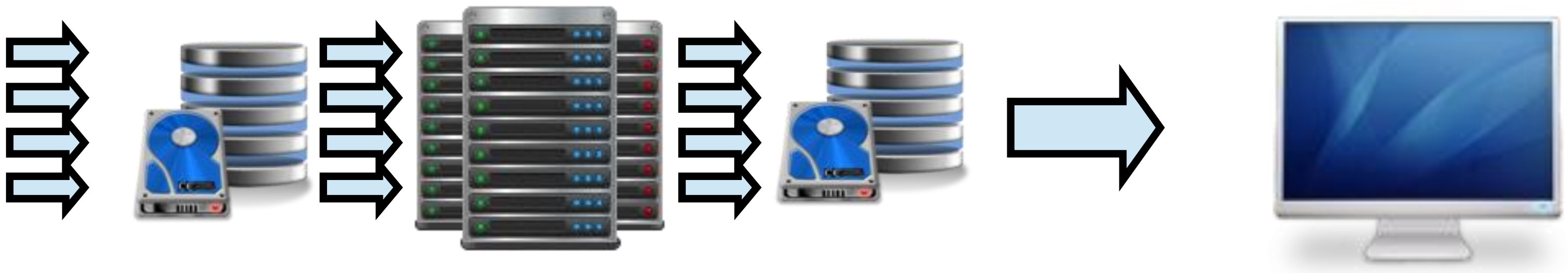
Multiple Independent Jobs

Examples:

- Parameter search
- Movie rendering

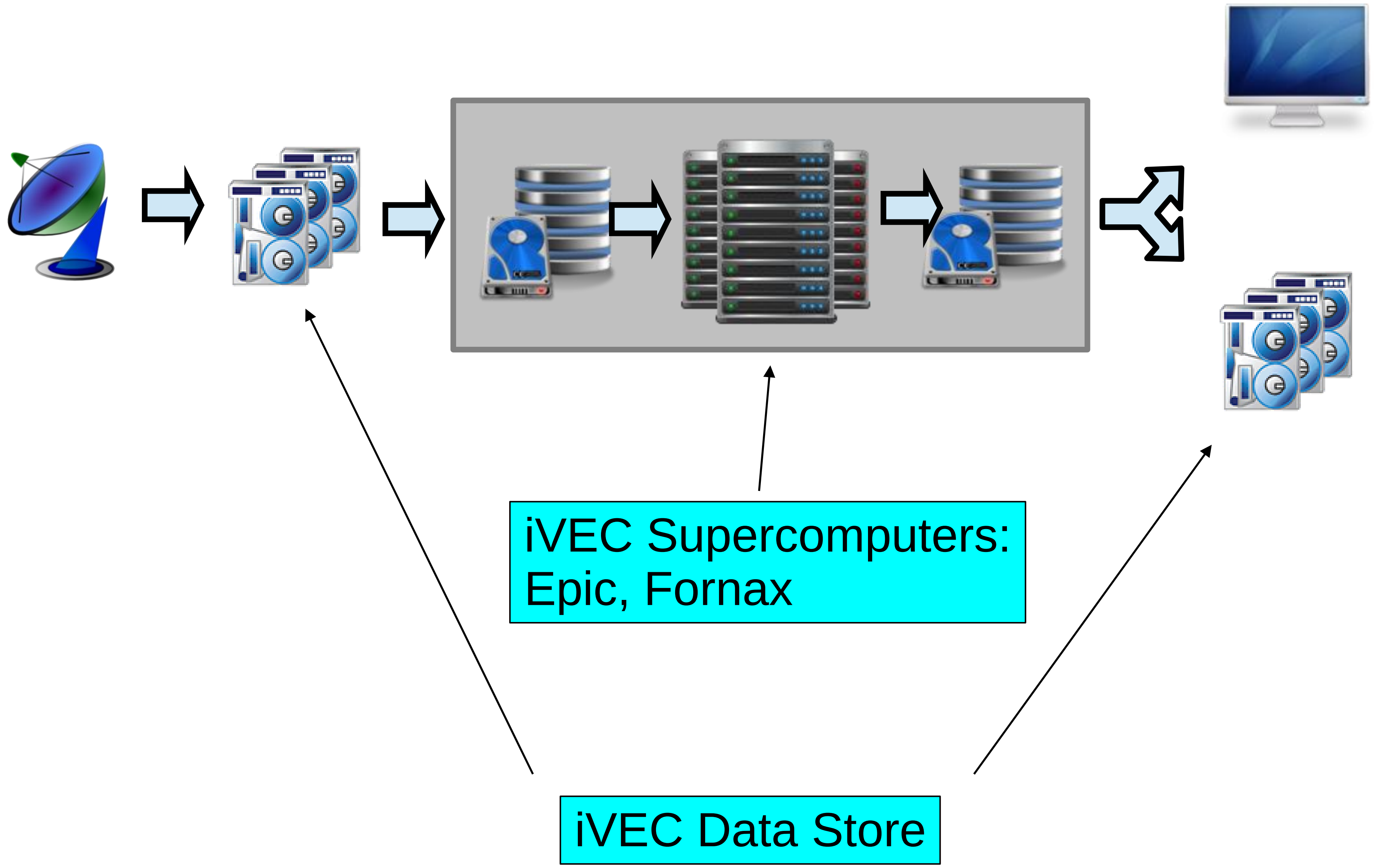
“Embarassingly parallel”.

Input file



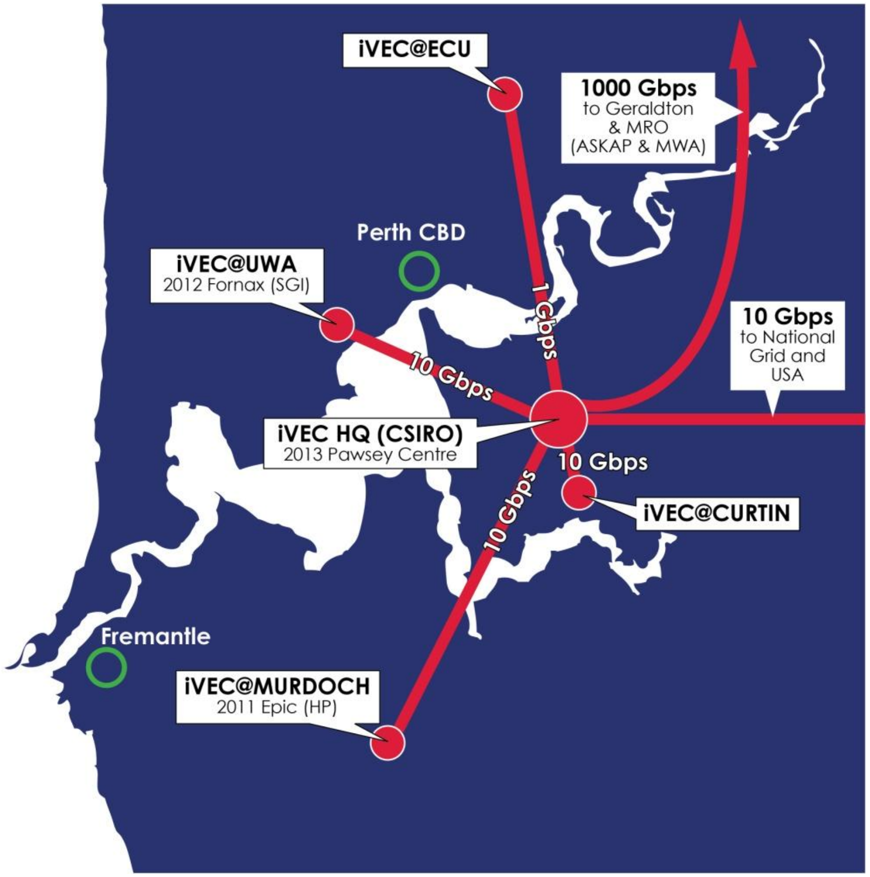
iVEC

Current iVEC Systems

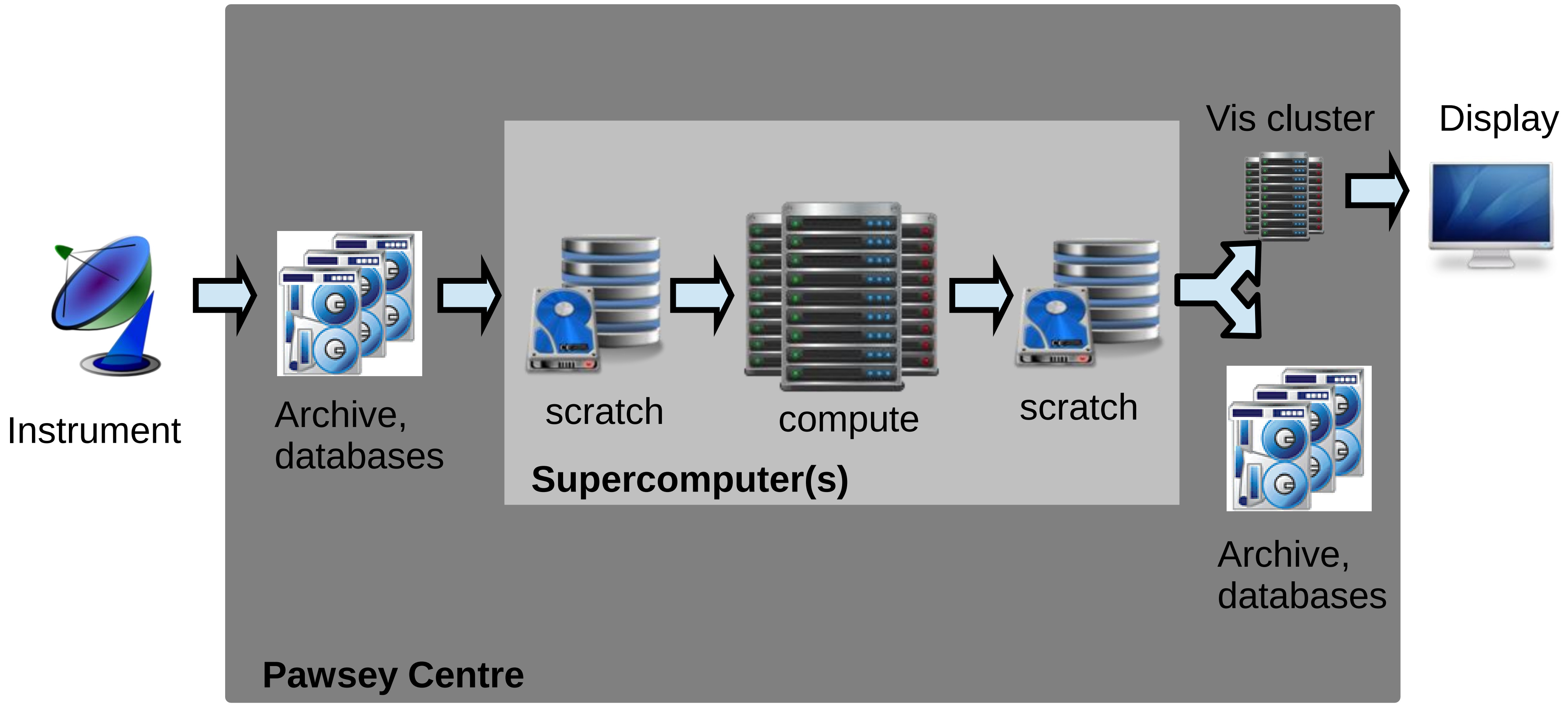


Distributed Data

- iVEC supercomputers and data store are in different locations, but joined by 10 Gbps network.
- Also joined to national supercomputers and data repositories at 10 Gbps (via AARNet).


iVEC@ECU
1000 Gbps to Geraldton & MRO (ASKAP & MWA)
Perth CBD
iVEC@UWA 2012 Fornax (SGI)
1 Gbps
10 Gbps
10 Gbps to National Grid and USA
iVEC HQ (CSIRO) 2013 Pawsey Centre
10 Gbps
10 Gbps
iVEC@CURTIN
Fremantle
iVEC@MURDOCH 2011 Epic (HP)

The Pawsey Centre



mid-2013 Onwards

II. SUPERCOMPUTERS

What Is a Supercomputer?

- Oxford Dictionary - “a particularly powerful mainframe computer”.
- Wikipedia - “A supercomputer is a computer at the frontline of current processing capacity, particularly speed of calculation”.
- Or – A computer that enables research that cannot be practically performed on a desktop computer.

Why Use One?

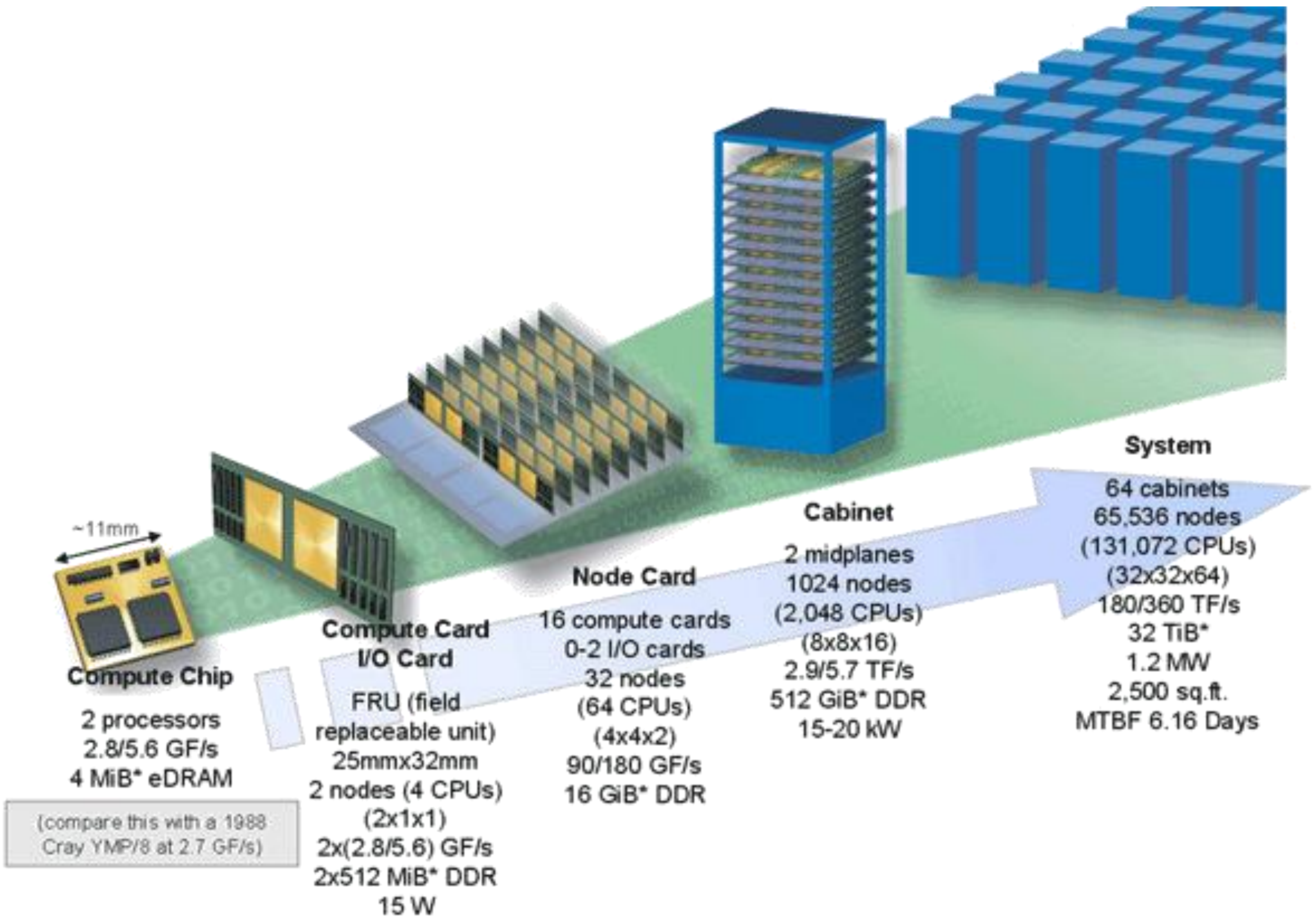
- Faster answers
- Bigger problems
- More scenarios
- Data throughput

Types of Supercomputers

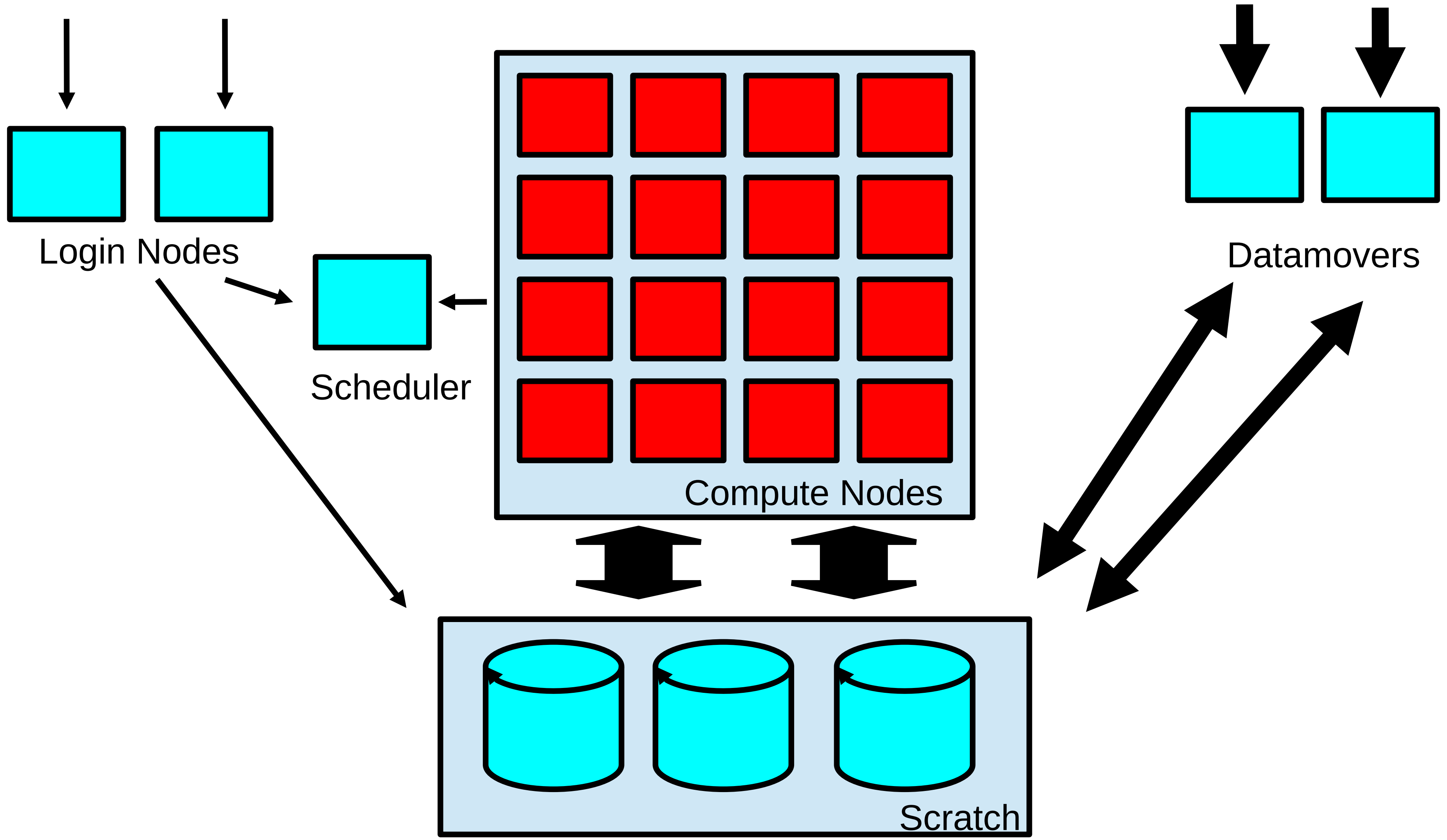
- *Distributed memory* – programs are split across multiple computers that are linked together. Each is called a node. A "cluster" is the most common of these supercomputers. A separate program runs on each node, and they have to explicitly communicate over the network to share data.

- *Shared memory* – the supercomputer looks like a single computer. (It might still be made from smaller computers linked together). A single program can access any of the memory in the supercomputer.

Example – IBM BlueGene



Abstract Supercomputer



Login Nodes

- For remote access to the supercomputer.
- Manage simulations.
- Many people share a login node at the same time.

<u>Don′t run your simulations on the login nodes!</u>

Login nodes can have different hardware to compute nodes.

- iVEC′s Fornax login nodes do not have GPUs
- Cray login nodes are not part of the main interconnect.

Compute Nodes

- Typically similar to commodity servers, with key differences:
  - Lower latency higher bandwidth network.
  - Improved thermal design. (Often water-cooled racks).
  - Sometimes have accelerators.
- Usually you do not share compute nodes while your jobs are running.
- Might have a slightly different operating system to the login nodes. E.g. Cray.

Scratch storage

Fast storage “inside” the supercomputer.

- Working area, not for permanent storage.
- Usually optimised for high throughput of data, not for low latency of small pieces of data.
- Could have thousands of nodes reading/writing at the same time.
- Scratch is shared. Multiple simultaneous users reduce performance.

Data Mover Nodes

Externally connected servers that are dedicated to moving data to/from scratch. You could use login nodes for this, but here are reasons to keep them separate:

- Data transfer buffers over long distances require large memory.
- Encrypted data requires CPU resources.
- Might have higher throughput network interfaces.
- Data movers are shared, but most users will not notice. (Depends on the other end, distance, encryption algorithm).

III. HARDWARE

Commodity Hardware

Supercomputer vendors leverage commodity hardware where practical. It means more compute for less money.

- Hardware in desktop computers (especially for gaming) is fast. R&D cost is spread across a large customer base.
- Server hardware leverages this, but with increased reliability. E.g. Intel Xeon vs Intel Core2 CPU, or DDR3 memory vs ECC DDR3 memory.

Cluster Supercomputers

- Clusters are a cheap way to build a supercomputer. Major factor is usually the network.
- Compute nodes based on servers – e.g. Intel Xeon CPUs, ECC DDR3 memory, PCIe bus for network and accelerators.
- Use lower latency higher throughput networks such as Infiniband, Cray Aries or SGI NUMAlink.
- Use optimised accelerators, not gaming GPUs – e.g. NVidia Tesla GPUs for double precision arithmetic.

Expensive Supercomputers

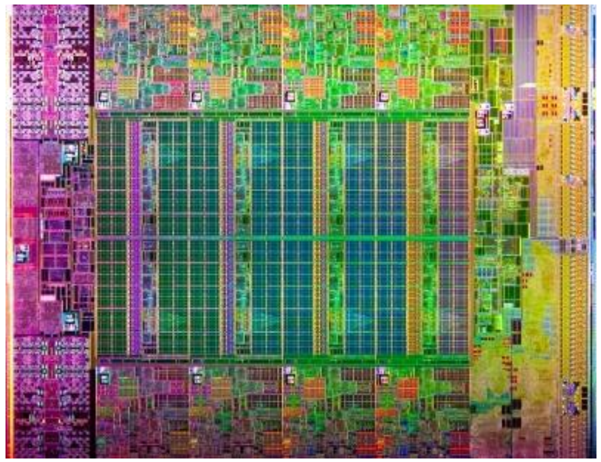
e.g. IBM BlueGene

- Memory soldered onto motherboard – hard to fix, but better resilience.
- PowerPC CPU – better power efficiency, but more expensive

CPUs

Central Processing Units.

- Complex cores – good for branching, task switching. Lots of silicon means more electricity.
- Lots of hardware for control.
- Multiple levels of cache.
- Typically around 6-10 cores.

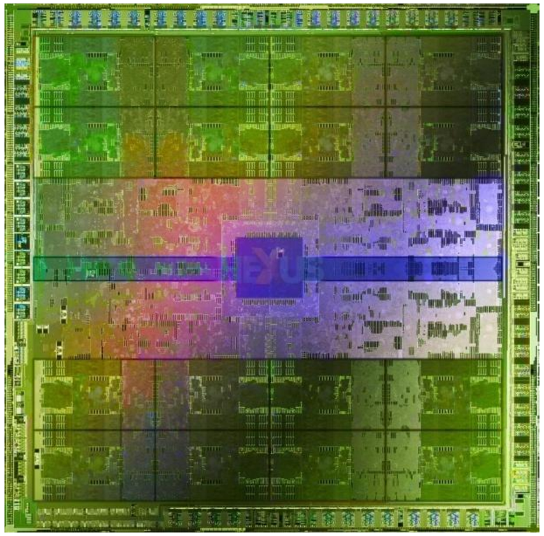


Intel E5-2600 CPU

GPUs

Graphics Processing Units. “Accelerated Processing Units”.

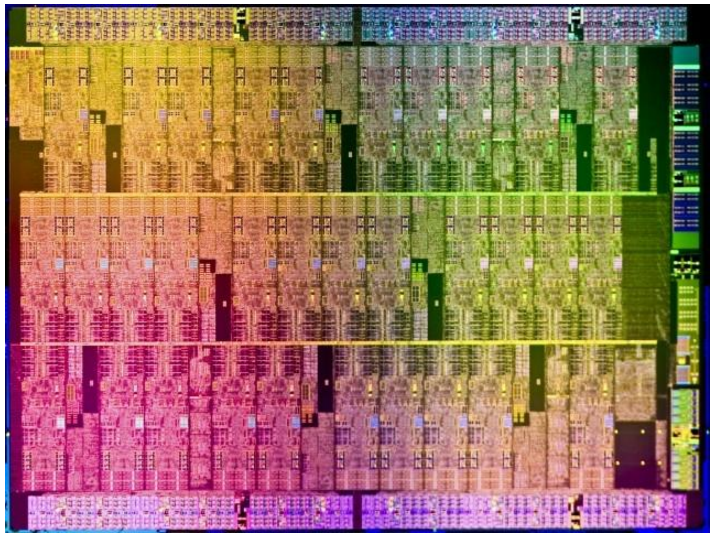
- Originated from gaming.
  - Lightweight kernels performing identical operations on many pixels or triangles.
  - Original focus on single-precision floating performance, but double-precision performance now good.
- Simpler control hardware.
- Efficient electricity usage.
- Need to move data over PCIe bus



NVIDIA
Fermi GPU

Intel Xeon Phi

- Intel's Many Integrated Core architecture
- Lots of simple x86 cores on a chip
- Relative newcomer to APU scene
  - Results yet to be shown for wide range of applications

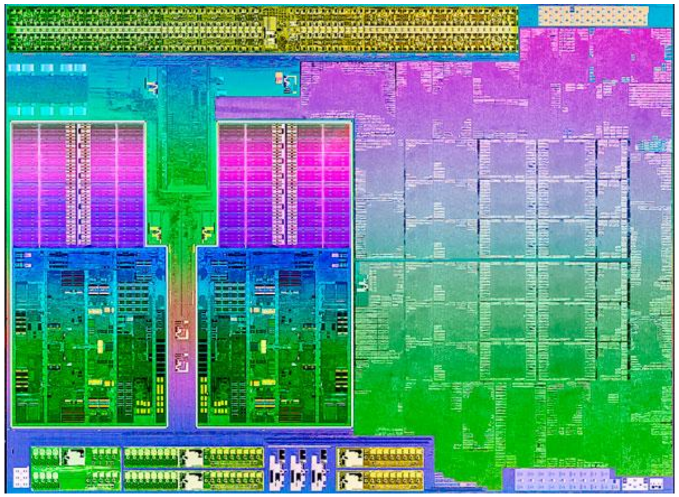

Intel Xeon Phi

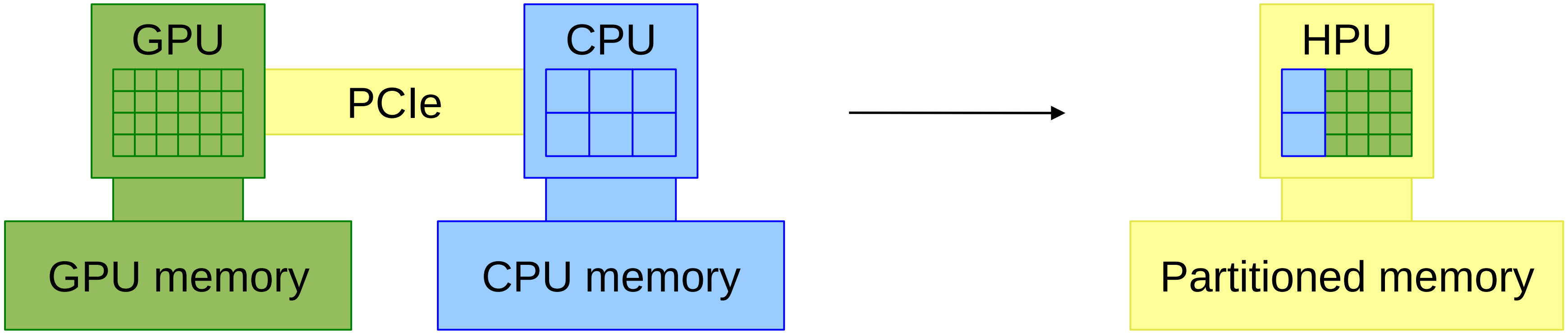
Heterogeneous Programming Unit

Merging of CPU and APU hardware.

AMD, Intel and Nvidia all doing it.

- Easier to program
- Bypass the PCIe bus
- Smaller computers
- Fewer components


AMD Trinity.



Network

How the nodes of the cluster communicate.

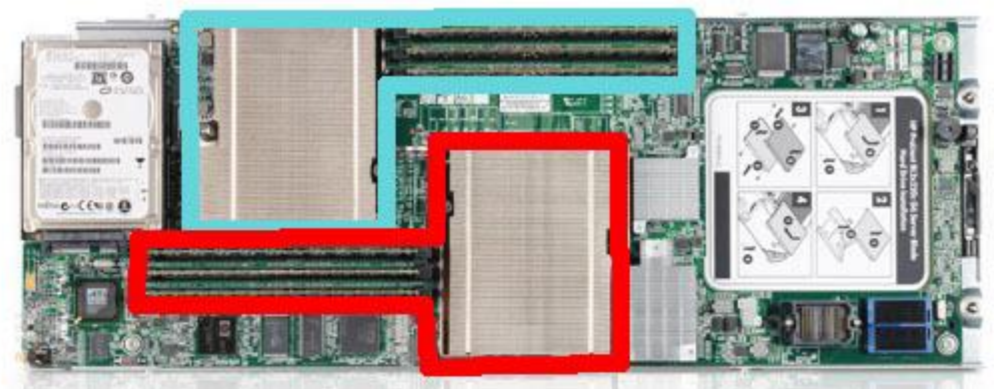
- E.g. Ethernet, Infiniband, Cray Aries.

Important factors are:

- Bandwidth. The rate you can transfer data.
- Latency. How long it takes for a small message to be received.

Epic

- Two hex-core Intel Xeon CPUs per node, with 12GB memory each.
- 800 blades connected by QDR Infiniband network.
- 9600 cores and 18TB memory.
- Lustre global filesystem.
- NUMA memory on blades, distributed memory between blades.


A HP ProLiant BL2x220c G6, with two CPUs

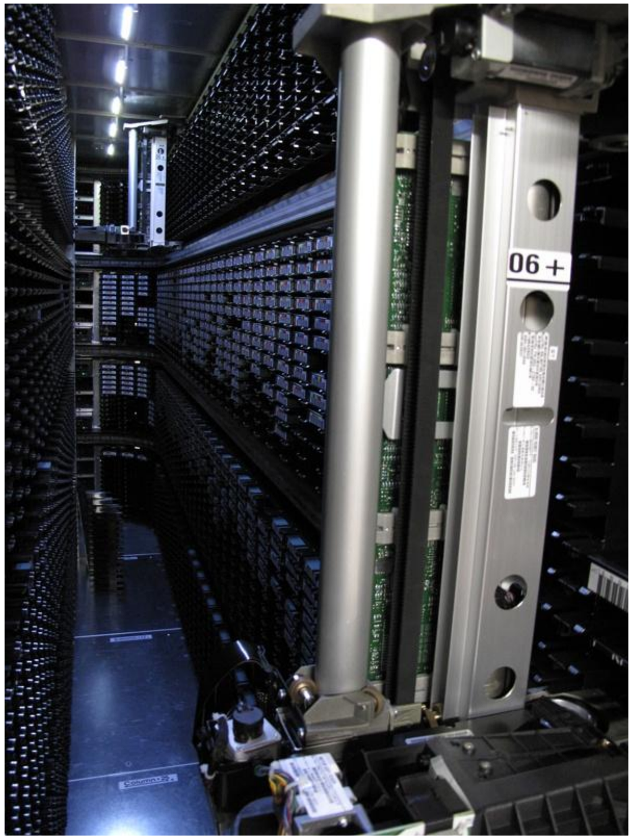
Fornax

- Two hex-core Intel Xeon CPUs per node, with 36GB memory each.
- One Nvidia Tesla C2075 GPU per node.
- 96 nodes connected by dual rail QDR Infiniband. (One for storage, one for programs).
- 500TB Lustre global filesystem
- Backed by DDN storage array.
- Also NUMA between CPUs.

iVEC Data Archive

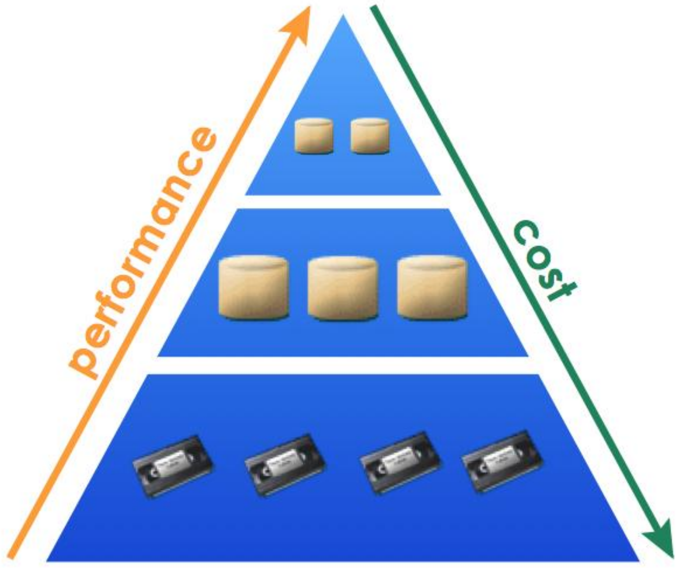
- Called the “iVEC Petascale Data Store” or “cortex”.
- HSM - hierarchical storage management. Tape library with hard disk cache.
- 6,500 tape slots, 800GB-5TB.
- Appears as a single filesystem.

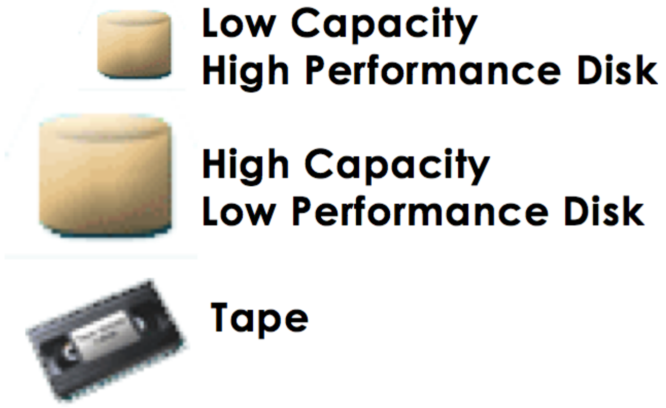
- Tape is not good for random access.
- Works best with large contiguous files.
- Can take minutes to access data.
  - (hard disks take milliseconds)

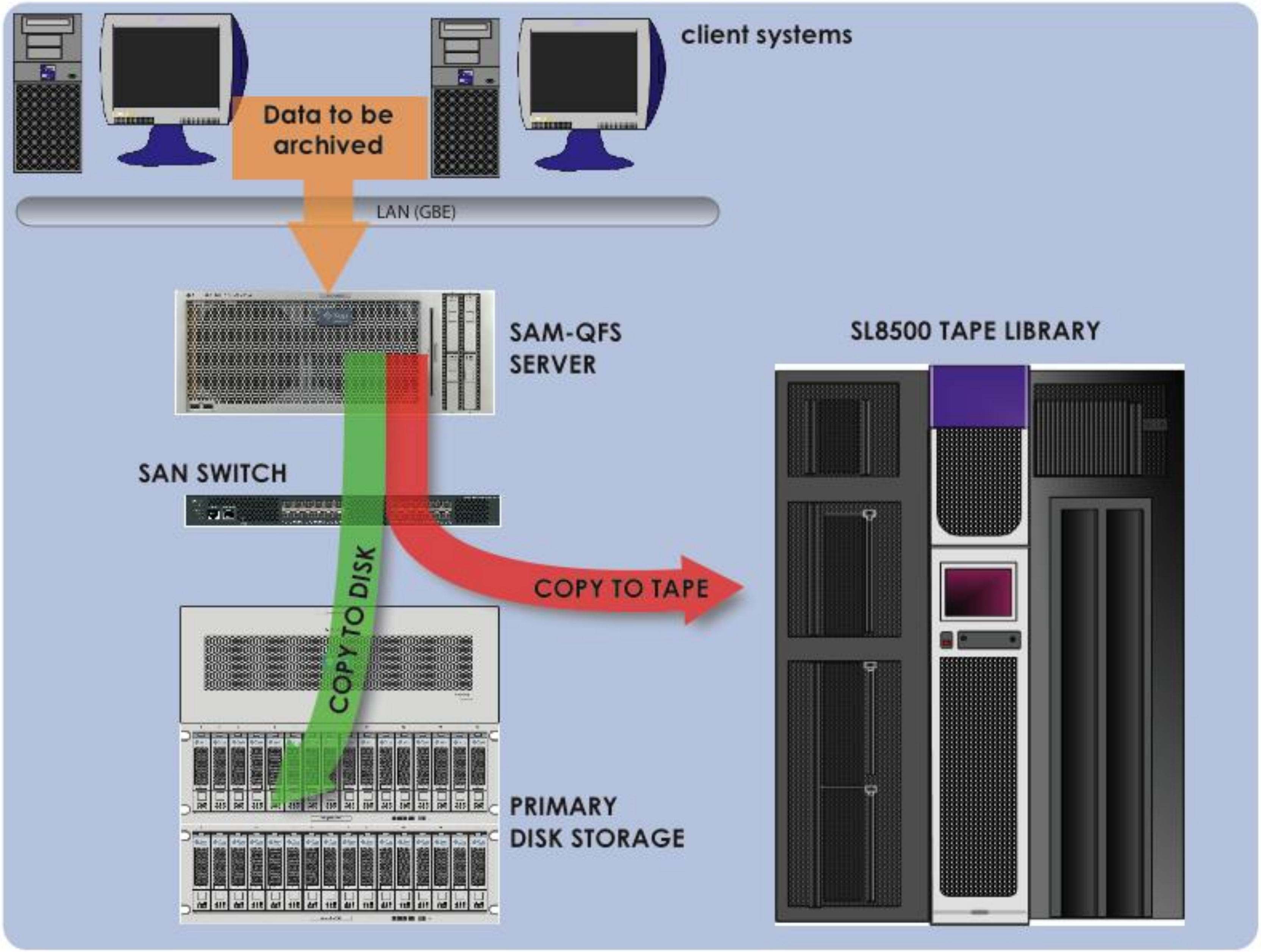


HSM? What is it?

- ***H**ierarchical **S**torage **M**anagement*
- Transfers data between tiers of lower cost storage
- Tiers – Disk Disk Tape or Disk Tape (even optical storage)
- Least accessed data is moved to a lower tier of storage






client systems
Data to be archived
LAN (GBE)
SAM-QFS SERVER
SAN SWITCH
COPY TO DISK
COPY TO TAPE
SL8500 TAPE LIBRARY
PRIMARY DISK STORAGE

iVEC Data Store

It is a data archive / repository!

Not for backups of source code.

- Use Sourceforge, Github, Google Code, code.ivec.org, etc.

Not for backups of your desktop/laptop.

Getting Data to the iVEC Data Store

Accessing the data store:

- Log in to cortex.ivec.org
- Move data via cortex.ivec.org

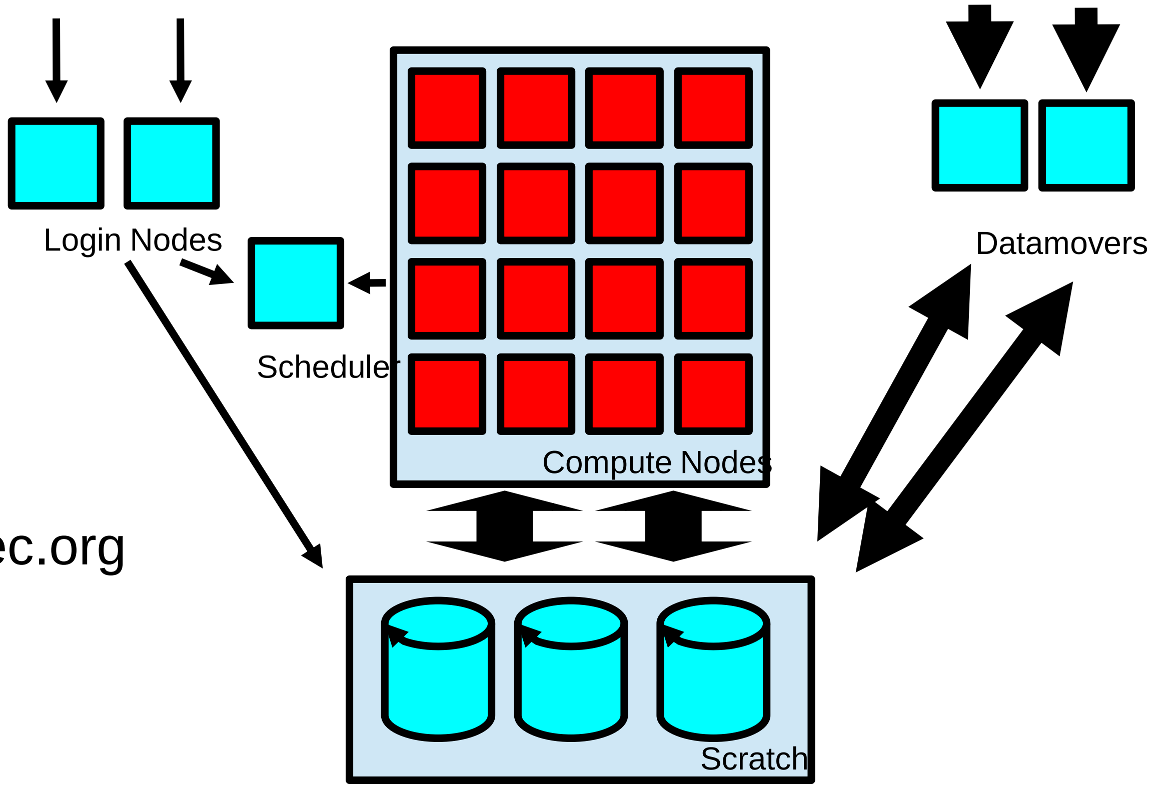
E.g.

tar cf ***myarchive***.tar ***mydatadirectory***

scp -rp ***myarchive***.tar ***username***@cortex.ivec.org:/pbstore/***myproject***/

Accessing iVEC Supercomputers

Standardised naming of the datamover nodes...



Epic supercomputer

- Log in to epic.ivec.org
- Move data via epicdata.ivec.org

Fornax supercomputer

- Log in to fornax.ivec.org
- Move data via fornaxdata.ivec.org

IV. MANAGING EXPECTATIONS

Expectations of a Supercomputer

Your experience of running parallel programs on a supercomputer is limited by:

- Theoretical limits
- Practical limits
- Other people

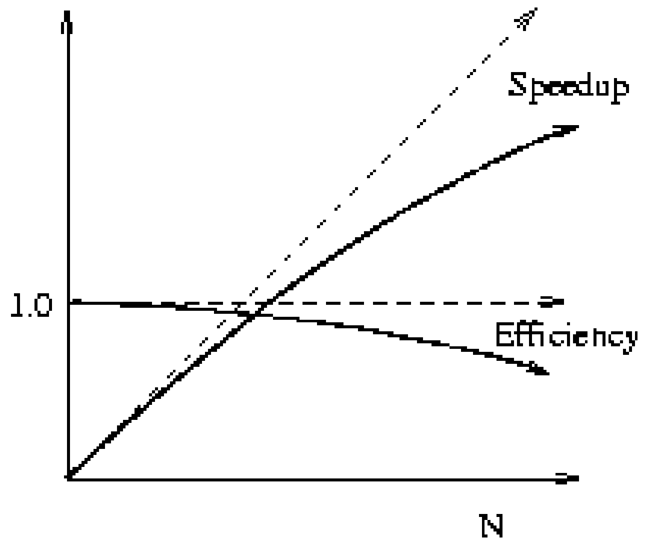
Amdahl's Law

Recall the cake baking exercise:

- Some tasks can be done in parallel.
- Some tasks must be done by one cook, or every cook must do the task. These are serial tasks.

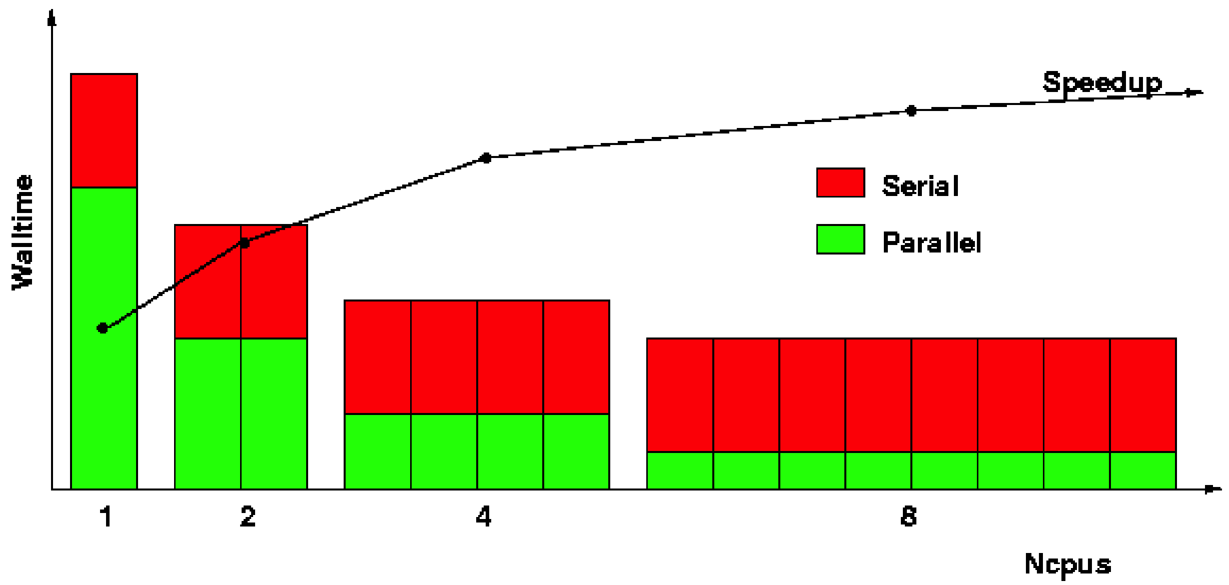
Measuring Performance

$$\text{Speedup(N cpus)} = \frac{\text{walltime(1 cpu)}}{\text{walltime(N cpus)}}$$



Efficiency(N cpus) = Speedup(N cpus) / N

Amdahl's Law – In Theory

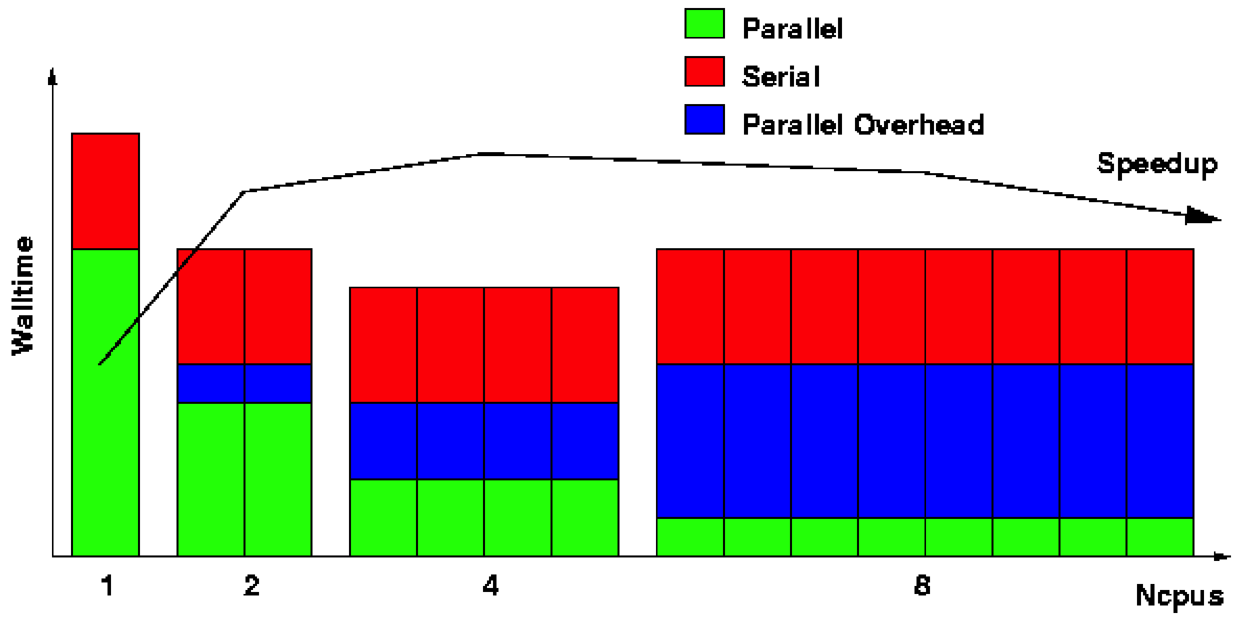


- More workers make the parallel part faster.
- Eventually limited by the serial tasks.

Discussion: Amdahl's Law

- What factors limit the parallelisation of baking of a cake?
- What is a sensible number of cooks?

Amdahl's Law: In Practice



- Parallel overhead (e.g. coordination / communication) eventually dominates.
- With too many workers it takes longer.
- There is a sweet-spot for the fastest time.

So How Many CPUs?!

Want to minimize both:

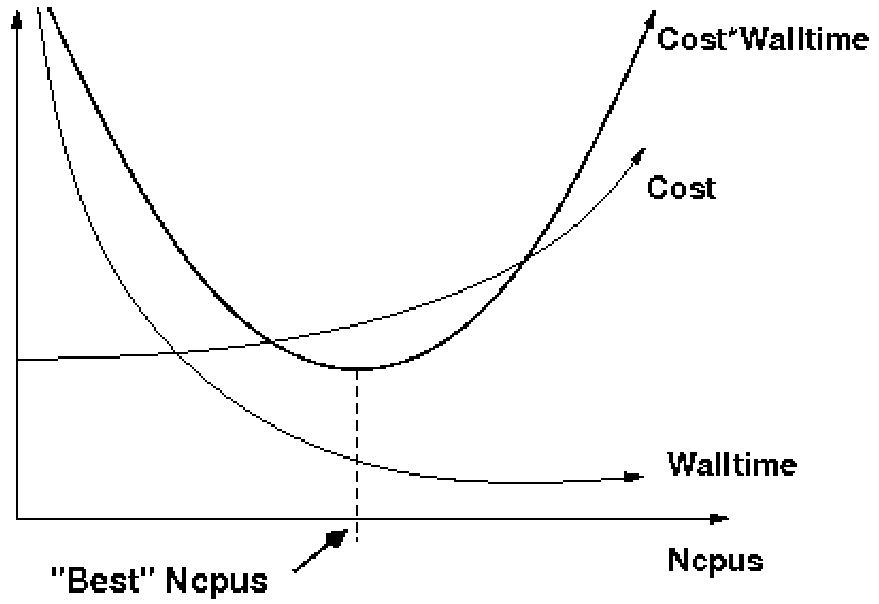
- •time to solution (walltime)
- •cost (= ncpus*walltime)

Contradictory!

How about minimizing Cost*Walltime?

Work out your own scheme.

Do some timing runs.



Algorithms

Many algorithms have:

- Increments of time
- Elements of interest
- Ways to visit these elements
- A calculation <do something>

Algorithm A - Grid

```
For t = t1 ... tn  {over time increments}
    For i = $x_1$ .. $x_n$
        For j = $y_1$ .. $y_n$
            For k = $z_1$ .. $Z_n$
                <Do something>
```

Algorithm B - N-body

```
For t = t1 ... tn  {over time increments}
    For each element
        For each other element
            <Do something>
```

Algorithm Scaling

At this stage, we can make a few observations:

A will do #t.#x.#y.#z calculations = $O(t*n^3)$

B will do #t.n.(n-1) calculations = $O(t*n^2)$

… this is an indication of the **order** of an algorithm.

Resource Requirements

Compute and memory can scale differently.

	Compute Time	Memory
3D-grid with time	$O(t*n^3)$	$O(n^3)$
n-body	$O(t*n^2)$	$O(n)$

What happens if we parallelise this?

Resource Requirements

Perfectly parallelised version:

- Compute is faster, memory per node is reduced.
- Now have communication

	Compute Time	Memory	Communication
3D-grid with time	$O(t*n^3)/p$	$O(t*n^3)/p$	??
n-body	$O(t*n^2)/p$	$O(t*n)/p$	??

Setting Expectations

Knowing the behaviour of the algorithm means you can set your expectations on how long the simulation will take, and how much resource you need.

E.g. if I make my 3D grid resolution 10 times finer, it will take 1000 times longer to run.

Often it makes more sense to use a better scaling algorithm than to try to parallelise a poorly scaling algorithm.

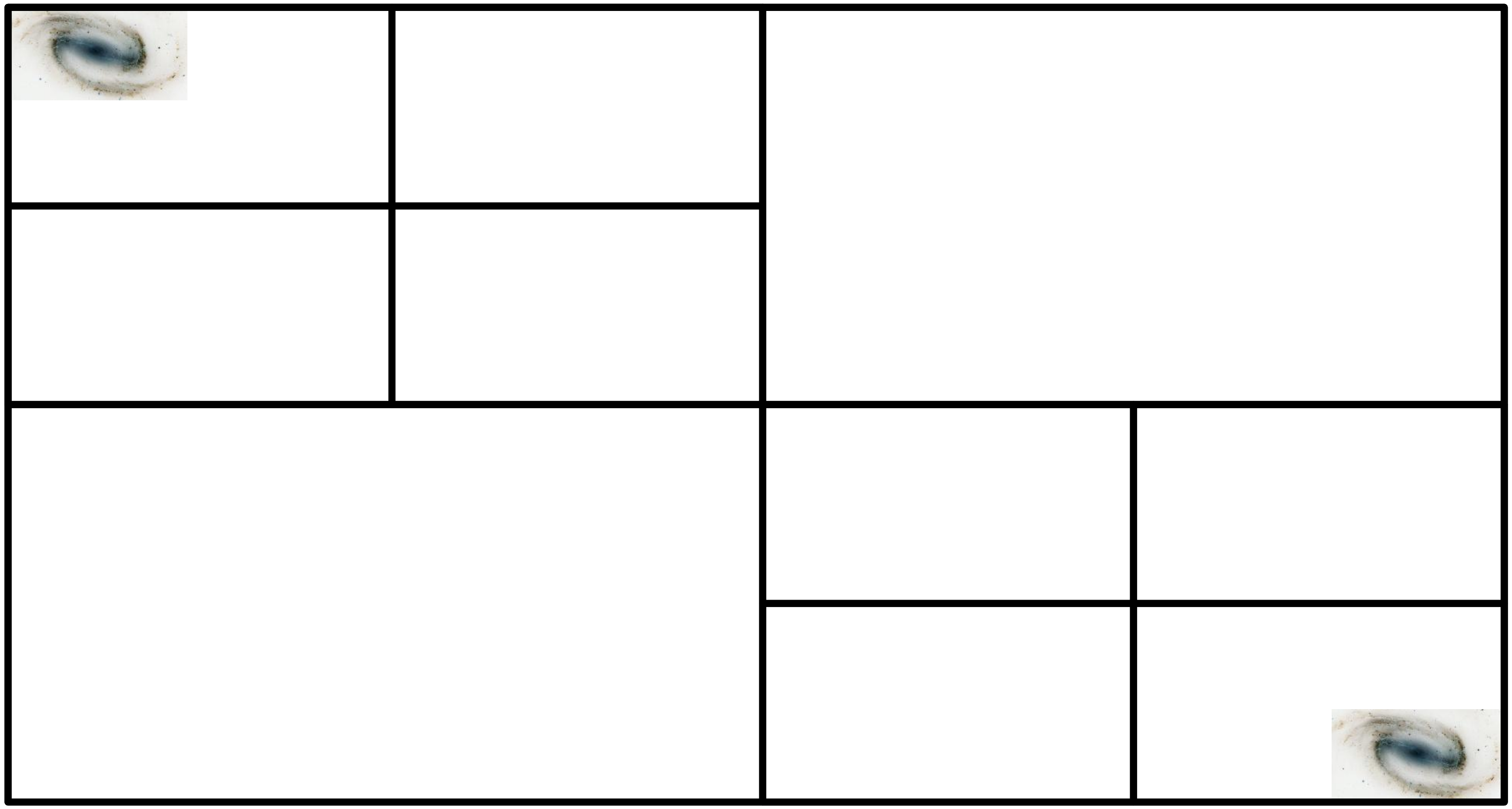
N-Body Example

Instead of calculating every pairwise interaction (brute force), represent hierarchies of bodies with multipoles.

$O(N^2)$ becomes $O(N \log N)$, even $O(N)$ for adaptive expansion.

N	Brute force $O(N^2)$	Multipole model $O(N \log N)$
1,000	1,000,000	7,000

Adaptive Spatial Decomposition



Strong vs Weak Scaling

Strong scaling means more processors can make a fixed problem run faster. E.g. ten times the processors make it run almost ten times faster.

Weak scaling means more processors can make a bigger problem run faster. E.g. ten times the processors means we can run a problem 10 times bigger in around the same amount of time.

Supercomputers enable big problems to be solved, but do not necessarily make small desktop simulations faster.

Granularity

Recall from the baking discussion:

- Fine-grain parallelism – low-level parallelism.
- Coarse-grain parallelism – high-level parallelism.

V. SHARING WITH OTHERS

Multitasking

Modern CPUs and operating systems can switch between tasks. It makes them feel snappy when responding to human input.

Multitasking lowers the performance of each individual task

- Switching to a different task means the cache contains irrelevant data for the new task.
- Minimise switching.

Make sure only one task runs on one core.

  – How do we control this on a supercomputer?

Dedicated resources

At iVEC you normally get compute nodes dedicated for yourself.

- This means other users will not task switch with you.
- It does not mean your own programs won't task switch.
- It is up to you to use the resource efficiently.

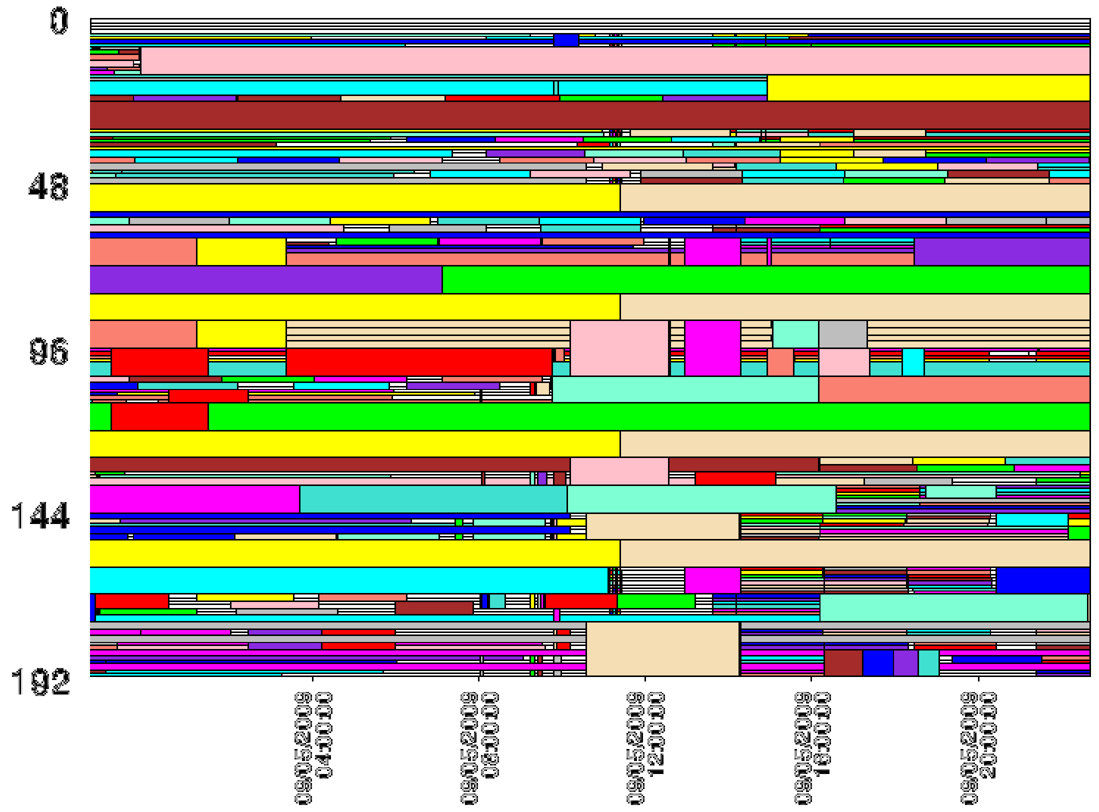
Batch queuing systems are used for dedicating nodes at iVEC.

Queueing Systems

- Researchers need dedicated resources to run most efficiently.
- iVEC needs to ensure the resources are utilised and not left idle.
  - All computers get replaced every 3-5 years and use electricity whether used or not.
  - Want them running at night and weekends.

Solution: have many people queue up jobs in advance, and a scheduler can optimise filling the machine.

A Scheduler in action



Filling a 192-cpu supercomputer over a day, with jobs of different sizes and lengths.

Scheduling Your Job

Tell the scheduler what resources your calculation needs.

Overestimating the time required means it will take longer to find an available slot.

Underestimating the time required means the job will get killed.

Underestimating memory will cause your program to crash.

Resource Charging

- At iVEC you do not pay directly for resources allocated.
- Get a yearly allocation, but you can exceed this.
- Your priority in the queue depends on how much allocation you have used, relative to everyone else in the queue. This is core to the “Fair Share” algorithm.

Don't waste your allocation!

- Don't have a thousand cores reserved and then spend the first few minutes moving data.
- Stage data before you need it, especially from tape.

Optimising Throughput (Outcomes)

It is easier for the scheduler to fit short jobs in.

- Break jobs up (checkpoint and restart)
- Beware of making jobs too small – if the machine is busy you could spend a lot of time sitting in the queue.
- iVEC queues have time limits (typically 12 hours).

The queuing system can be used for parallelisation.

- Run small simulations independently rather than making each go faster. This minimises parallel overheads.
    - E.g. Parameter sweeps
    - Can use job arrays

VI. GETTING HANDS-ON

iVEC Queues

All iVEC supercomputers use PBS Pro to manage queues.

To list the queues when logged into a machine:

```
qstat -Q
```

To display more information about a queue:

```
qstat -Q -f queuename
```

Epic Queues

There are multiple queues to choose from on epic:

routequeue	feeds workq, smallq (default)
largeq	>128 cpus, 12 hr limit
mediumq	>12 cpus, <=128 cpus, 12 hr limit
smallq	<=12 cpus, 12 hr limit
longq	<=12 cpus, 12-96 hr limit, 48 GB nodes
copyq	for copying large amounts of data; datamover nodes
debugq	small, quick test runs (96 cpus/1 hr limit)
testq	for iVEC staff only

Fornax Queues

There are multiple queues to choose from on fornax:

workq	< 95 nodes per job, 24 hr limit (default queue)
longq	< 4 nodes per job, 120 hr limit
copyq	for copying large amounts of data; datamover nodes
courseq	for training courses
debugq	small, quick test runs (1 node/4 hr limit)
develq	development nodes on Fornax; have different hardware
testq	for iVEC staff only

Which queue?

On some machines such as at the National Facility, different queues have different charging weights.

e.g. an express queue with higher priority might be charged higher than a standard queue.

For training on Epic and Fornax, we will use courseq

Exercise – Log in

```
ssh myusername@fornax.ivec.org
cd /scratch/courses01/myusername

cp /group/courses01/intro_to_supercomputing/* .
```

Jobscripts

PBS Pro executes a script submitted to it. Must be bash, csh or python. You cannot submit a program directly.

PBS Pro interprets directives at the top of the script, which are written as comments.

Example Jobscript

```
#!/bin/bash
#PBS -N JobName
#PBS -W group_list=courses01
#PBS -q courseq
#PBS -l walltime=00:01:00
#PBS -l select=4:ncpus=12:mpiprocs=12:mem=23GB
#PBS -l place=free:excl
#PBS -m ea
#PBS -M my@email.address.edu.au

cd $PBS_O_WORKDIR

module load intel
module load openmpi

printenv

mpirun ./myprogramname
```

PBS flags

#PBS -N ***JobName*** – makes it easier to find in qstat

#PBS -W group_list=***courses01*** – project accounting

#PBS -m ea – send notification email at job end or abort

#PBS -M ***my@email.address.edu.au*** – where to email job status

#PBS -l walltime=***00:01:00*** – request the resource for this long

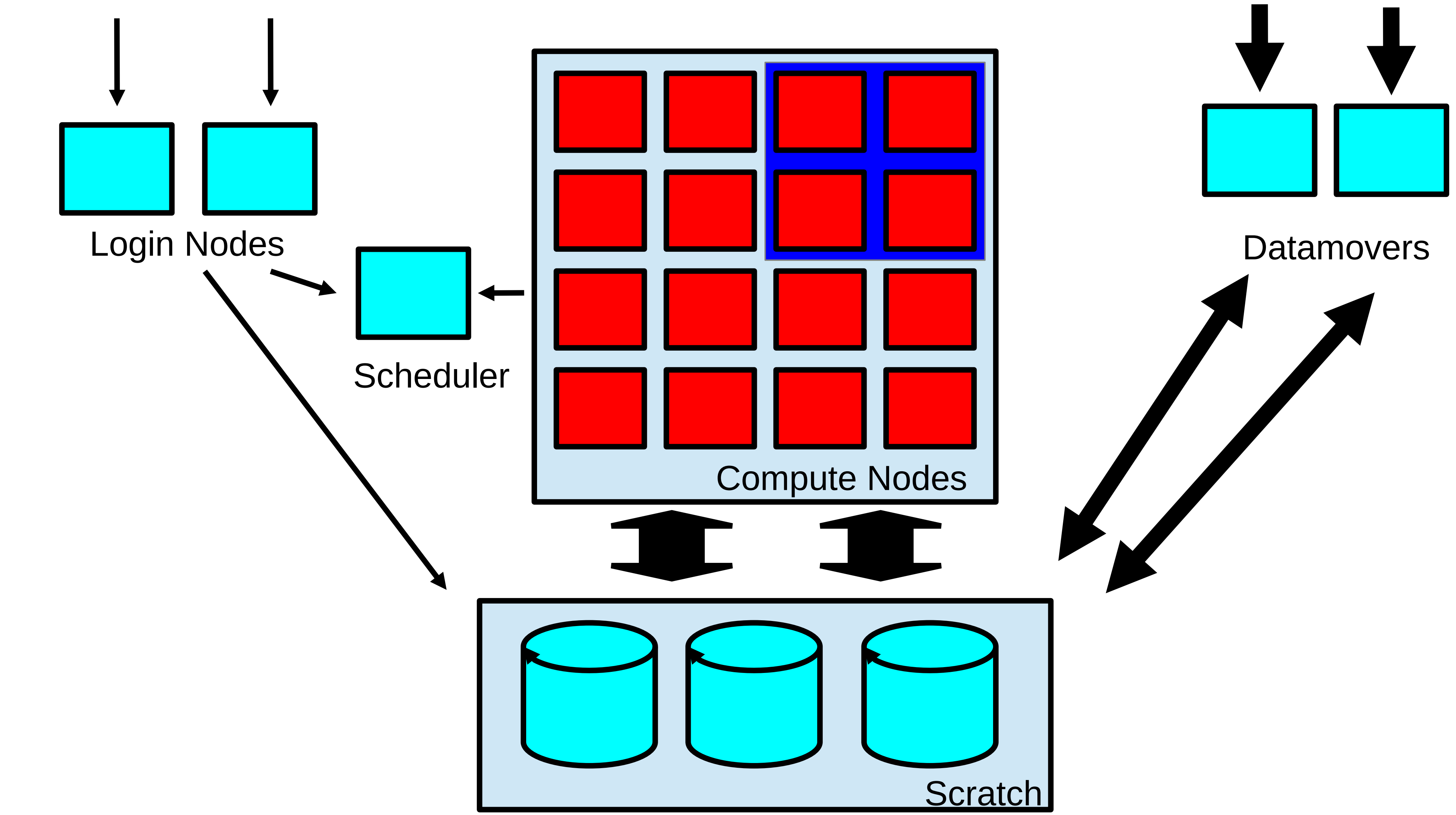
Select Notation

On clusters, to get a large number of CPUs or memory you need to use multiple compute nodes.

select notation asks the scheduler for a number of chunks with certain properties. If you think of chunks as compute nodes, then it is straightforward. The number of chunks to ask for is the number of nodes that get you how many CPUs or memory you need.

This means you need to know the details of the compute nodes of the supercomputer and queue.

A job spanning 4 nodes



Select Notation

#PBS -l $\underbrace{\text{select=}\textbf{\textit{4}}}$:$\underbrace{\text{ncpus=}\textbf{\textit{12}}\text{:mpiprocs=}\textbf{\textit{12}}\text{:mem=}\textbf{\textit{23}}\text{GB}}$

1st part is the number of chunks requested.

2nd part is the specification of each chunk.

- In PBS Pro ncpus is the number of cores.
- On Epic, each node has 12 cores, 24GB RAM.
- On Fornax, each node has 12 cores, 72GB RAM.
- On Fornax also use ngpus=1 for a GPU.
- Don't forget the operating system requires memory!

PBS Pro and Python

Put PBS directives in as comments near the top, the same as for shell scripts.

```
#!/usr/bin/python -O
#PBS -W group_list=courses01
#PBS -l walltime=00:01:00
#PBS -l select=1:ncpus=1:mem=1GB

print “hello world”
```

PBS Job Dependencies

Add this near the top of your jobscript:

#PBS -W depend=afterok:***123456.epic***

The job will not start until job ***123456.epic*** has completed sucessfully.

- This is very useful in conjunction with a job in the copy queue.
- Use after instead of afterok if it does not matter whether the previous job was successful (such as for resubmission scripts).

Useful queue commands

qsub ***scriptfilename***

qstat

qdel ***jobid***

You get told the identifier ***jobid*** when you qsub the job.

Querying the Queue

qstat displays the status of jobs in the queue.

qstat –u ***username***

The "S" column is the status. Usually see one of:

- Q – job is waiting in the queue
- R – job is running
- H – job is held.

PBS Output

Standard output and standard error from your jobscript are collected by PBS, and written to files in the directory you submitted the job from when the job finishes/dies.

scriptname.o***jobidnumber***

scriptname.e***jobidnumber***

If you used the -N **name** flag, replace scriptname above with the ***name***.

These files also contain useful PBS information.

Useful PBS commands

q – short for “qstat –u $USER”

qtotal – shows useful information about your jobs in the queue.

They are on both Epic and Fornax.

Modules

Modules modify the environment to easily locate software, or particular versions of software.

module avail

- List all available software

module list

- List the software versions (modules) you are currently using

module show ***modulename***

- Shows information about the module.

module load ***modulename***

module load ***modulename/version***

module unload ***modulename***

Modules Example

```
> printenv LD_LIBRARY_PATH
    -- empty --
> printenv PATH
    /usr/local/bin:/bin:/usr/sbin
> module load intel
> module load openmpi
> printenv LD_LIBRARY_PATH
    /ivec/devel/intel/12.1/openmpi/1.6.3:/ivec/intel/12.1.3/compiler/lib/intel64:/ivec/intel/12.1
      .3/mkl/lib/intel64
> printenv PATH
    /ivec/devel/intel/12.1/openmpi/1.6.3/sbin:
      /ivec/devel/intel/12.1/openmpi/1.6.3/bin:/ivec/intel/12.1.3/mkl/tools:/iveci/intel/12.1.3/
      bin/intel64:/usr/local/bin:/usr/bin
```

Compiling code

You need to “module load” a compiler before you can compile a program.

We recommend the Intel compilers (most of the time) on Intel-based systems. They support Fortran, C, C++ and OpenMP.

On Epic and Fornax the intel module is loaded by default.

On Fornax often you will need the CUDA library:

```
module load cuda
```

Exercise – compiling LAPACK

Get and unpack the source code:

```
> wget http://www.netlib.org/lapack/lapack-3.4.2.tgz
```

Unpack it:

```
> tar xzf lapack-3.4.2.tgz

> cd lapack-3.4.2
```

Exercise – compiling LAPACK

Customise the compiler options:

> cp make.inc.example make.inc

Edit make.inc;

```
FORTRAN = ifort
LOADER = ifort
NOOPT = -O0 –fltconsistency
TIMER = INT_CPU_TIME
```

Compile it in a jobscript in the right directory:

```
make blaslib
make –j 12
```

Compiling MPI code

OpenMPI has wrapper programs for the Intel compilers, which take care of locating and linking the MPI libraries.

```
module load intel
module load openmpi
mpif90 –O2 –o myprog.exe a.f90 b.f90 c.f90
```

“mpif90 --show” will show what the wrapper does.

Exercise – Compiling with MPI

```
module load intel
module load openmpi
mpif90 -O2 -o darts-mpi.exe darts-mpi.f
mpirun ./darts-mpi.exe
```

Try running darts-mpi.exe both with and without mpirun.

Exercise – Compiling with OpenMP

```
module load intel
ifort –openmp -O2 -o darts-omp.exe darts-omp.f

export OMP_NUM_THREADS=4
./darts-omp.exe
```

Intel Math Libraries

These are used often, as they include the BLAS, LAPACK and FFT libraries. It consists of multiple libraries – use the Intel advisor to work out the compiler link options:

http://software.intel.com/sites/products/mkl/MKL_Link_Line_Advisor.html

Example output:

```
-L$(MKLROOT)/lib/intel64 -lmkl_intel_lp64 -lmkl_sequential -lmkl_core
   –lpthread -lm
```

$(MKLROOT) is set by “module load intel”

Notes on iVEC Login Nodes

Epic login nodes have low process limits due to the high number of users sharing them.

- scp to epicdata.ivec.org, not epic.ivec.org.
- Don’t have too many login sessions.
- Run interactively in the queue with “qsub -I”.

Fornax login nodes do not have GPUs. Some compilation tests may fail (such as if you use autoconf).

- Run interactively in the queue with “qsub -I”.

Running Programs

A program usually requires libraries that were used while compiling it. Use appropriate “module load” commands in the job script so it can find those libraries. This includes the Intel compilers.

E.g.

```
module load openmpi
module load scalapack
module load petsc

mpirun ./myprogramname
```

Running programs

Distributed memory programs

- The program must be started on every node.
  - mpirun does this for mpi programs. It gets the job layout from PBS Pro automagically.

  mpirun ***./myprogram***

Shared memory programs (OpenMP, CUDA)

- Just run it normally.
  - May need to specify configuration, e.g. OMP_NUM_THREADS.

  ./myprogram

Hybrid programs (MPI + OpenMP, CUDA)

- The program is started on every node with mpirun.
  - May need to specify configuration, e.g. OMP_NUM_THREADS.

  mpirun ***./myprogram***

Modules on Fornax and Epic

Some modules have prerequisites and order is important.

```
> module load intel (or gcc)
> module load openmpi
```

Some modules are loaded by default. These can be unloaded.

- pbs
- intel
- openmpi

Exercise - PBS

You should have a file hostname.pbs with these contents:

```
#!/bin/bash
#PBS –W group_list=courses01
#PBS –l walltime=00:05:00
#PBS –q courseq
#PBS –l select=2:ncpus=2:mpiprocs=2:mem=68GB
#PBS –l place=free:excl

module load intel
module load openmpi

printenv
mpirun /bin/hostname
```

Exercise – PBS continued

qsub hostname.pbs

Use qstat to see it in the queue. If the queue is empty the job may run too quick to see.

Examine the output of the .o and .e files from PBS. How do the hostnames relate to the select= syntax?

Use “qstat –f ***jobid***” on a long-running job in the queue.

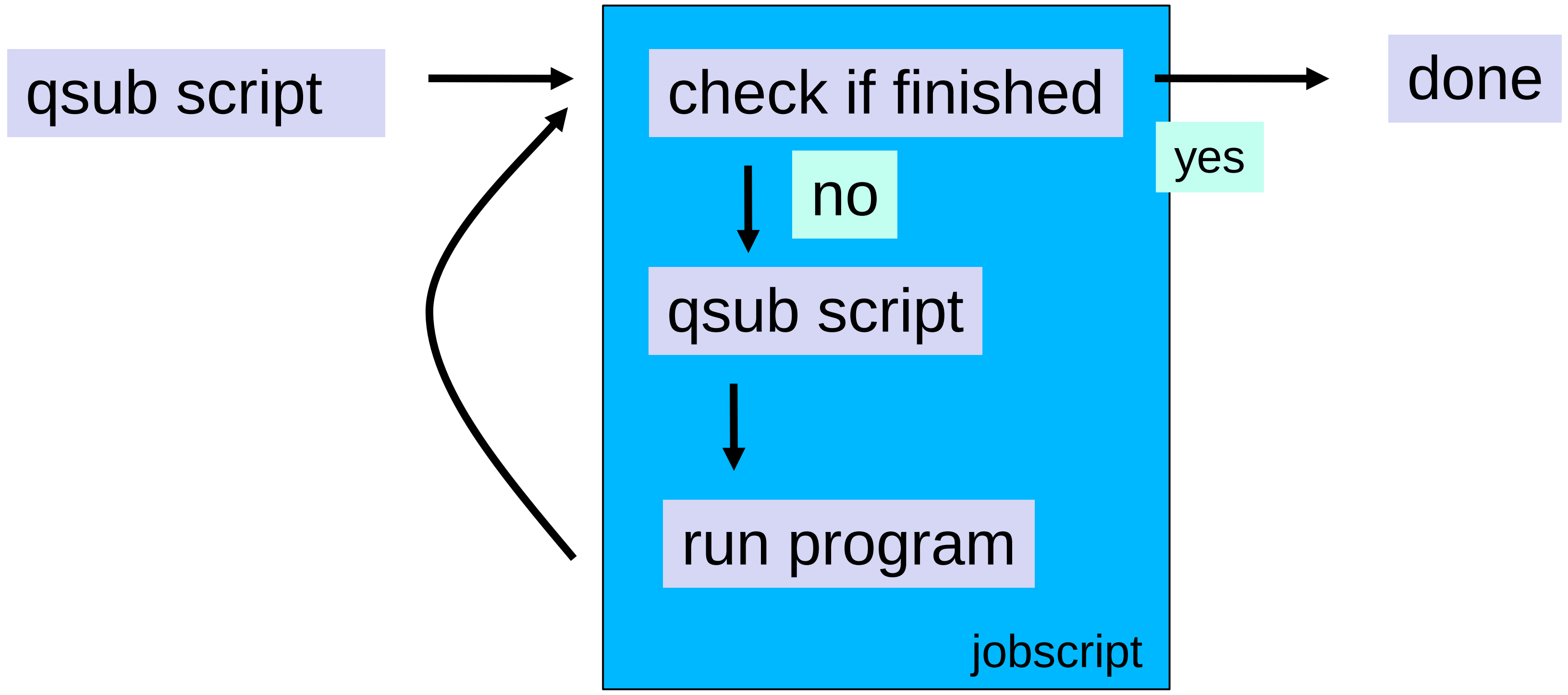
Checkpoint / Restart

The queues have time limits that are deliberately short.

- More efficient scheduling means greater throughput.
- Supercomputers have many components, so mean time between failure (MTBF) significantly reduced.

- Save the state of the program regularly.
- Queue a job to start from where it last saved.
  - You can use job dependencies to do this.
  - qsub -W depend=afterany:***epic.12345***

Resubmission scripts



Staging Data

Getting data into the supercomputer could take minutes or hours.

Use job dependencies and the copyq.

- No idle compute nodes during the transfer.
- Job starts automatically when the data is ready.

Staging Data

This combines job dependencies and using multiple queues.

Step 1. copyq job runs on the data mover node, pulling data into the supercomputer, using one shared CPU.

Step 2. defaultq job runs on dedicated compute nodes. Use “qsub -W depend=afterok:***epic.12345***” notation.

Step 3. copyq job runs on the data mover node, pushing data off the supercomputer. Again use “qsub -W depend=afterok:***epic.12345***”.

Note depend=afterok, not depend=afterany.

Staging data with scp

In the copyq use a unique passwordless ssh key.

Do not make your normal ssh key passwordless!

Do not use it on any other system.

Lock it down as best you can. In .ssh/authorized_keys file, prepend the line with

```
from=”*.ivec.org”,no-port-forwarding,no-X11-forwarding,no-agent-forwarding,no-pty
```

Email help@ivec.org to help set it up. It is a big security risk to your data if you get it wrong. If in doubt, don't do it.

Know Where to Put Things

Typical Filesystems on a shared cluster

Home directory – small, possibly slow, not backed up

Globally shared – slow or fast, global, not backed up

Scratch / JobFS – fast, local, not backed up

Remember “not backed up”.

Supercomputer Filesystems

epic:/home

fornax:/home

- Use your home directory to store code and input files, but not large results. Small quota. “lfs quota /home”.
- Most results can be regenerated if you have the input files
- Not backed up

- Your directory is /home/***username***.

Supercomputer Filesystems

epic:/ivec

fornax:/ivec

- This is for user applications installed/managed by iVEC.
- Most software available using the module command is installed here.

Supercomputer Filesystems

epic:/group

fornax:/group

- /group is where shared project files should go. For example executables, data sets.
- Large quotas “lfs quota /group”

- Your directory is /group/***projectname***.

Supercomputer Filesystems

epic:/scratch

fornax:/scratch

- /scratch is where temporary files should go
- Small quotas, try not to keep data on there too long. “lfs quota /scratch”
- If it fills up in the middle of a calculation, you may just be out of luck

- Your directory is /scratch/***projectname***.

Exercise – Backing Up

Copy the the exercise files back to your laptop.

Use a scp client to fornaxdata.ivec.org.

scp -r **username**@fornaxdata.ivec.org:/scratch/courses01/**username** .

Windows users - WinSCP

Useful commands

On a given iVEC supercomputer:

```
> module load ivecclustertools
```

```
> ivecquota
```

- Display your usage and quota on /home and /scratch. It's a wrapper for “lfs quota”.

```
> ivecfreenodes
```

- Display how busy each of the queues are.

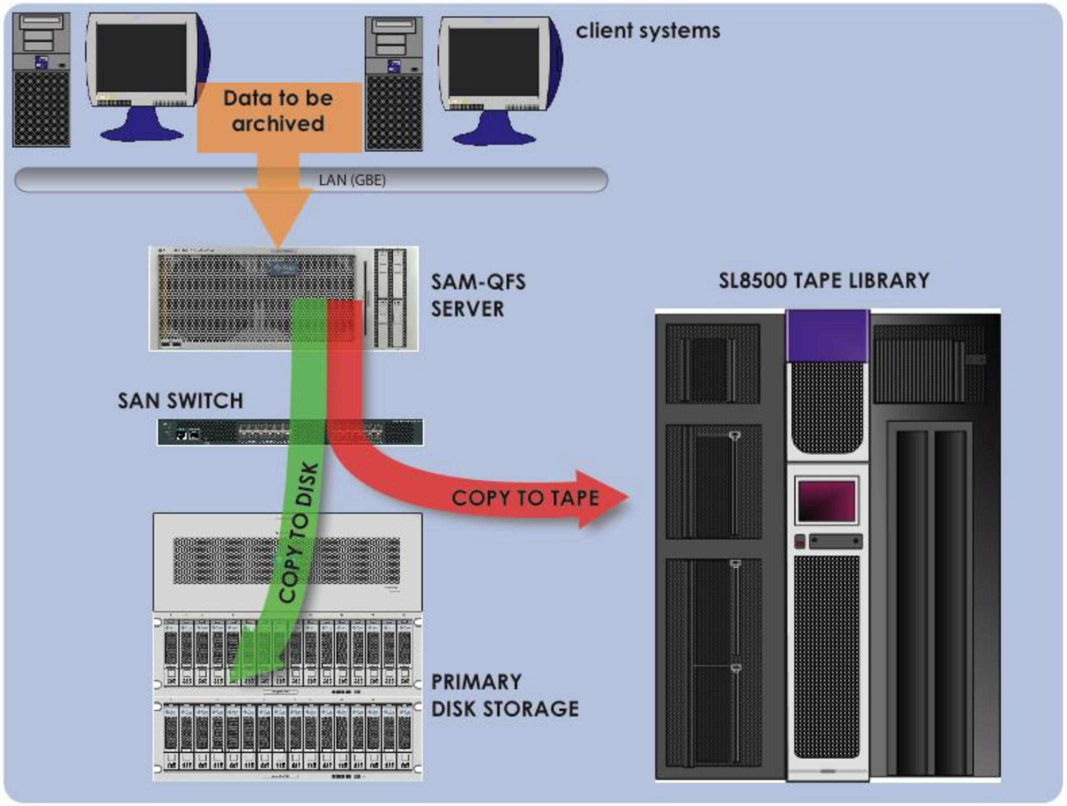
Moving Data

- Use secure authentication (encrypted)
  - GridFTP. GridFTP has multiple streams, works well over long distances.
  - SFTP/SCP. Is fine for small amounts of data.
- Aim to keep data close to the compute.
- “Stage” the data onto the supercomputer before you need it.

iVEC Data Store

Remember:

- At iVEC the data is always written to tape.
- When not used for a while, the disk copy is deleted.



Using the iVEC Data Store

- Most HSM systems use the same workflow and have similar commands.
- At iVEC, ssh to cortex.ivec.org and use the SAM-QFS tools to move and/or interrogate your files.

For completeness only:

- SAM - “Storage Archive Manager” provides the functionality of a Hierarchical Storage Manager.
- QFS - “Quick File System” is a multi-writer global file system, allowing multiple machines to read from & write to the same disks concurrently.

SAM QFS Tools Overview

archive - archive files to tapes

stage - retrieve files from the tape to disk cache

sls - similar to “ls”, with more file migration information

sfind - SAM-QFS “find”

release - release the file from disk caches

ssum - set file checksum attributes

sdu - “du” replacement size of archived directory/file.

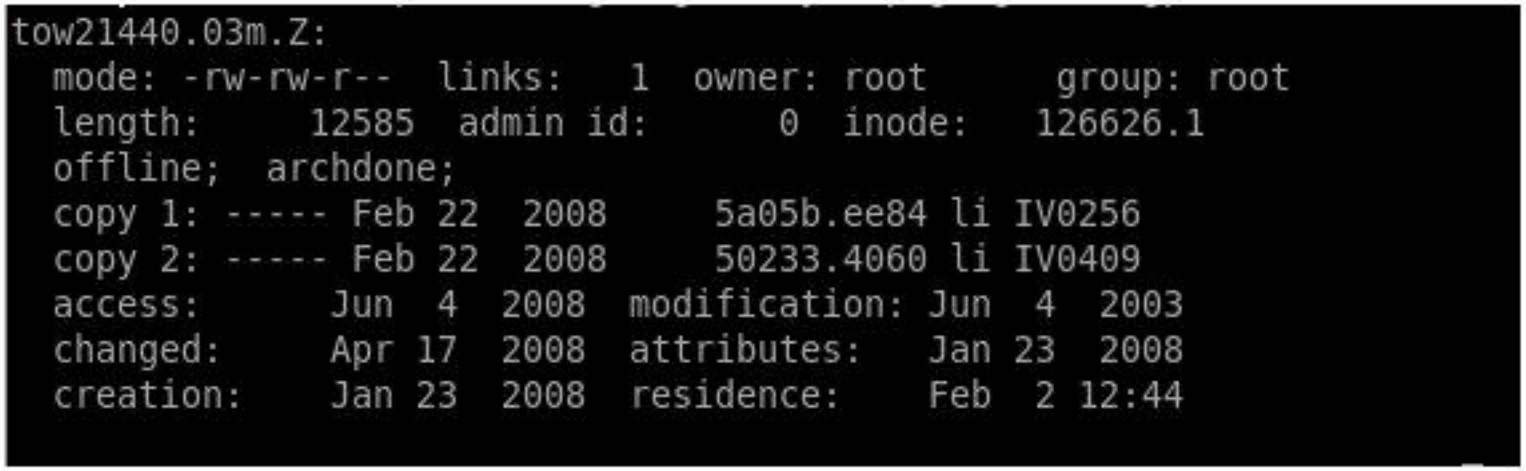
sls Output

damaged: the file is damaged.

offline: the file is offline.

archdone: indicates that the archiver has completed processing the file.

copy: Indicates the position of the file archive and the byte offset for each copy and the tape volume it is on.



```
tow21440.03m.Z:
  mode: -rw-rw-r--  links:   1  owner: root      group: root
  length:     12585  admin id:      0  inode:    126626.1
  offline;  archdone;
  copy 1: ----- Feb 22  2008     5a05b.ee84 li IV0256
  copy 2: ----- Feb 22  2008     50233.4060 li IV0409
  access:      Jun  4  2008  modification: Jun  4  2003
  changed:     Apr 17  2008  attributes:   Jan 23  2008
  creation:    Jan 23  2008  residence:    Feb  2 12:44
```

VII. ADMINISTRATION

IVEC User Portal

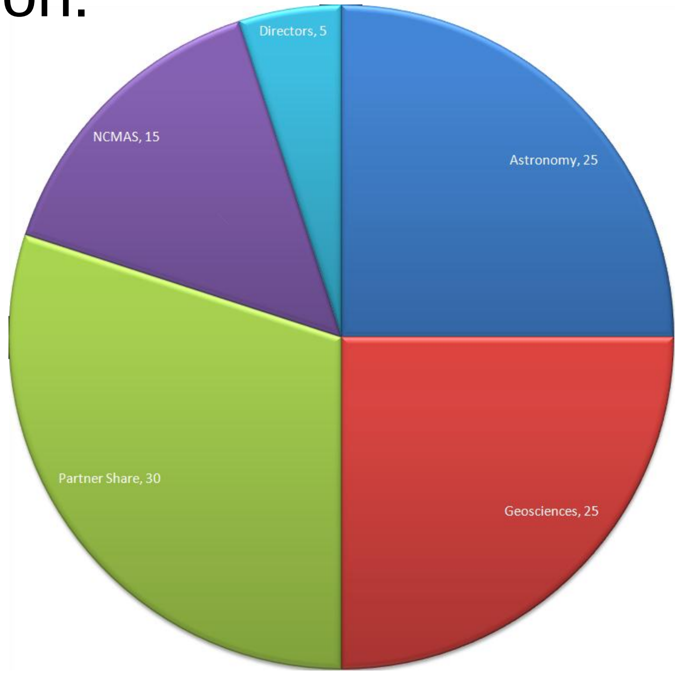
http://portal.ivec.org

- Documentation
- iVEC-supported Software list
- Maintenance logs

Applying for Access

- iVEC Supercomputers via Merit Allocation.
  - Astronomy (25%)
  - Geosciences (25%)
  - National Researchers (15%)
  - iVEC Partners (30%)
  - Director's Share (5%)

Call for applications around November.



- iVEC Data Store is via the iVEC helpdesk
- iVEC Visualisation equipment is via the iVEC helpdesk

Merit Allocation

- Highly competitive (over-subscribed).
- Depends on both research outcomes and effective use of iVEC resources.
  - Problems that **require** a large supercomputer preferred.
  - Codes that scale well are preferred.
  - GPU-based codes preferred on Fornax.
- You can apply through more than one allocation scheme. (E.g. National Partner and iVEC Partner).

Director's Share

- For researchers new to supercomputing.
  - Hand-holding from staff in the iVEC Supercomputing Technology and Applications Program.
- Expected to use the Merit Allocation Scheme afterwards.
- Ask us for help in filling out the form.

Other Available Resources

The National Partner share is allocated under the National Computational Merit Allocation Scheme. Other resources are too.

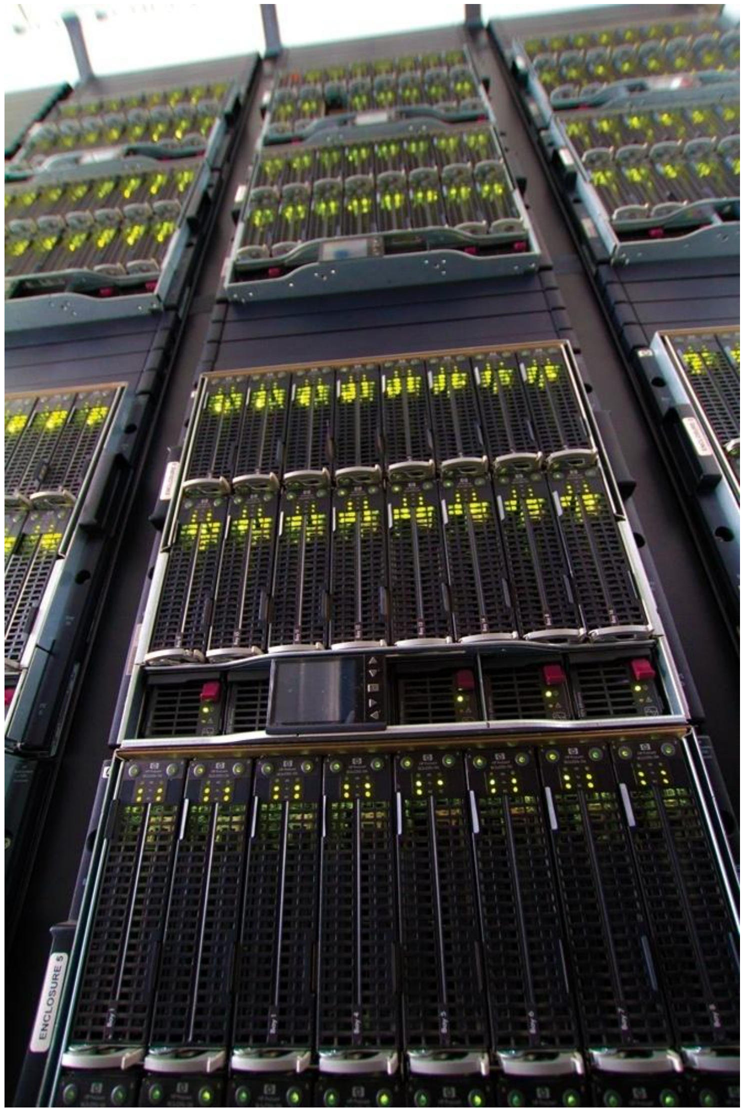
- NCI National Facility (CPU)
- iVEC Epic (CPU) and Fornax (GPU)
- NCI Specialised Facility in Bioinformatics (CPU)
- MASSIVE (GPU)

See http://ncmas.nci.org.au/

Supercomputing Technology and Applications Program

- Get more from supercomputing
- STAP-team expertise:
  - Access to iVEC facilities
  - High-performance computing
  - Parallel programming
  - Computational science
  - GPU accelerators
  - Cloud computing
  - Scientific visualisation
  - Data-intensive computing

help@ivec.org



iVEC Helpdesk

- Email help@ivec.org.
- There is no iVEC Helpdesk telephone hotline.
- Operates during Western Australian business hours.

More details at:

- http://portal.ivec.org/docs/The_iVEC_Helpdesk

iVEC Helpdesk

Help us to help you. Provide details!

E.g.

- Which iVEC resource.
- Error messages.
- Location of files.
- PBS job identifier.
- Your username and IP address if having login issues (but not your password!).