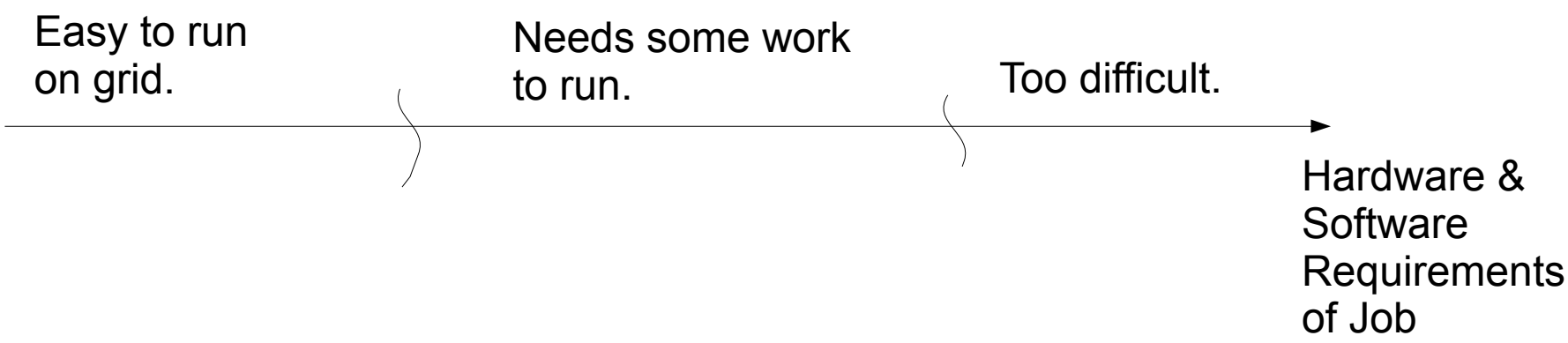
Using Opportunistic Resources

Marko Slyz
OSG User Support

Types of User Programs



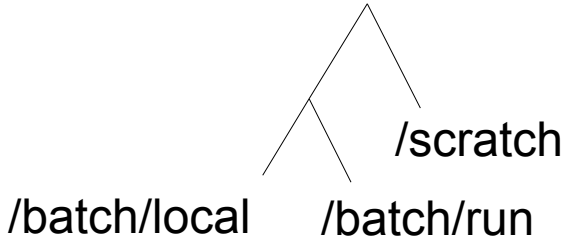
Checklist of Requirements

- Portability
- Job length/Workflow complexity
- RAM use
- Scratch disk use
- Data transfer to/from worker node
- Ways to recover failed jobs
- Ability to debug

Portability

The application

- should include any non-standard libraries or programs that it needs
- should be able to run anywhere in the file tree



- should not write to its install area if that's OSG_APP
- should be able to handle long paths.

Job Length/Workflow Complexity

Longer jobs are more likely to be preempted or evicted on opportunistic resources, so

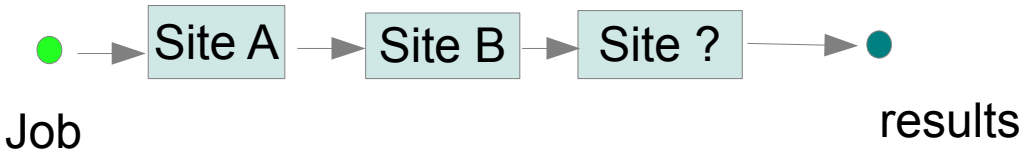
- Split up jobs if possible.

Use DAGMAN for dependencies. Disadvantages: More data to transfer, more things could go wrong.

- Run on long queues.
- Try to run at less busy sites.

Finding Quiet Sites (Experimental)

Method 1: Decide where a job should run next based on just its own recent history.



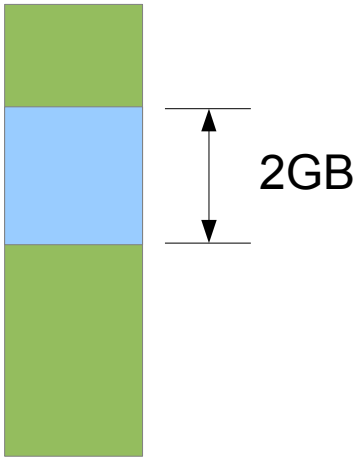

```
Rank  = (TARGET.Rank) - \
      ((TARGET.GLIDEIN_ResourceName =?= MachineAttrGLIDEIN_ResourceName0) * 1000) - \
      ((TARGET.GLIDEIN_ResourceName =?= MachineAttrGLIDEIN_ResourceName1) * 1000) -  ...
```

Method 2: Decide based on where other recent jobs have run well.

RAM Use

Should be about 2 GB ideally. Otherwise job can use swap too much or cause problems for other users, so check this beforehand.

/usr/bin/time prints ***maxresident*** *(but version 1.7 has a bug), or run ps every few minutes.*



Local Disk Use

Nominally 10GB per job should be available.

Could ship back and erase output data as it becomes available. Erase unneeded intermediate files.

Data Transfer and Storage

Basic use cases:

- small amounts of data → Condor file transfer

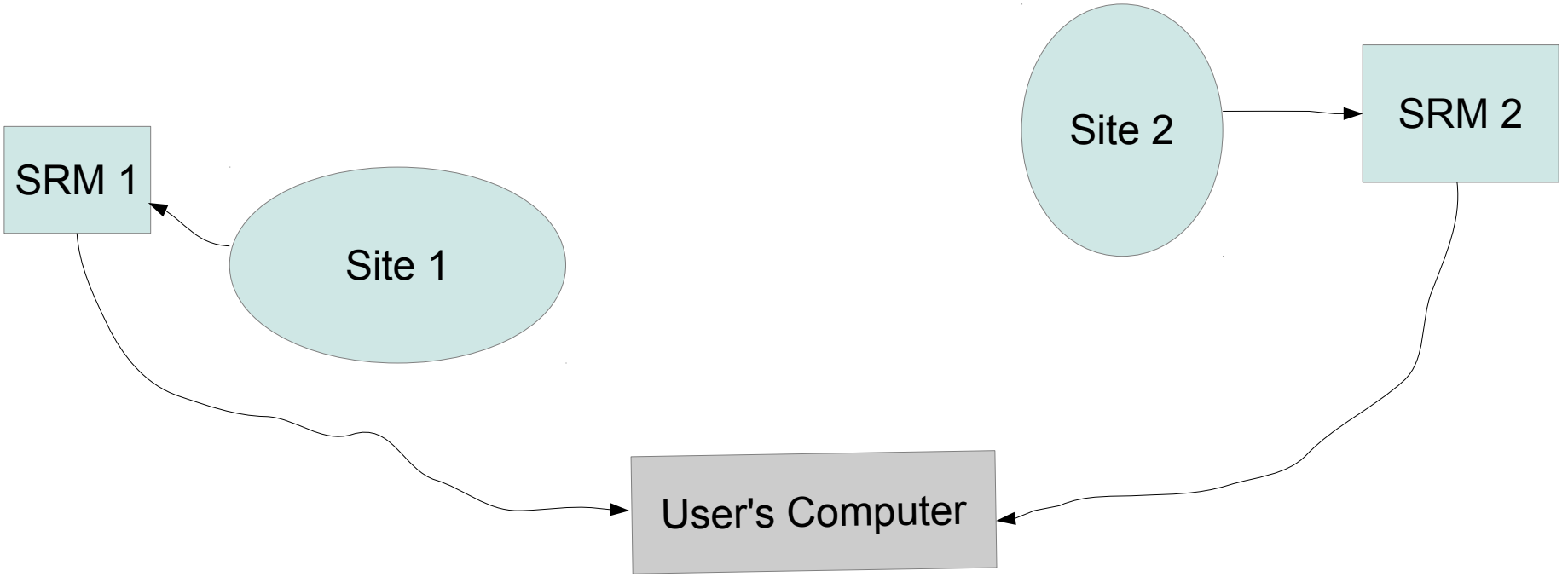
  Suggest 200 MB in, 1 GB out maximum.
- constant or staged data → shared file system at OSG_APP/OSG_DATA, or site gridftp/SRM server

  Can run job to do initial upload ("pull") or do transfer directly with gridftp/SRM ("push").
- Large amounts of non-constant data → have each job use SRM or gridtfp

Be aware of load on the network, especially for shared file systems.

A Prototype Data Transfer Workflow

For one user we saved the output each worker node produces–2 or 3 GB–into the site's SRM server. Then the user downloads all of the output–~600 GB in one case–with an SRM client.

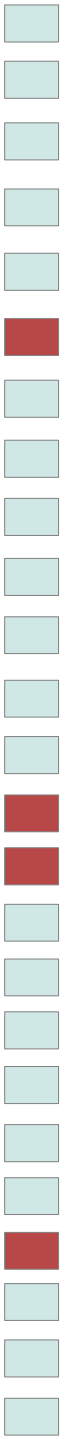


Recovering Failed Jobs

Job failures are normal.

Have a solution to detect and restart failed jobs.

Consider this before running production.



Debugging

Generate and return logs from program.

Use remote debugging tools if needed.

condor_ssh_to_job, glidein_ls, glidein_interactive

Return information about worker node.