Integrating Xgrid into HENP distributed computing model

Jérôme LAURET (Brookhaven National Laboratory)
Adam KOCOLOSKI (MIT)
Michael MILLER (MIT)
Levente HAJDU (Brookhaven National Laboratory)

Motivation & Outline

- Premise: Xgrid's single-vendor model simplifies many of the challenges associated with the management of a distributed computing cluster. Interfacing Xgrid with standard grid tools offers a unique method for resource harvesting and increases the diversity of hardware and software on which grid computing is possible.

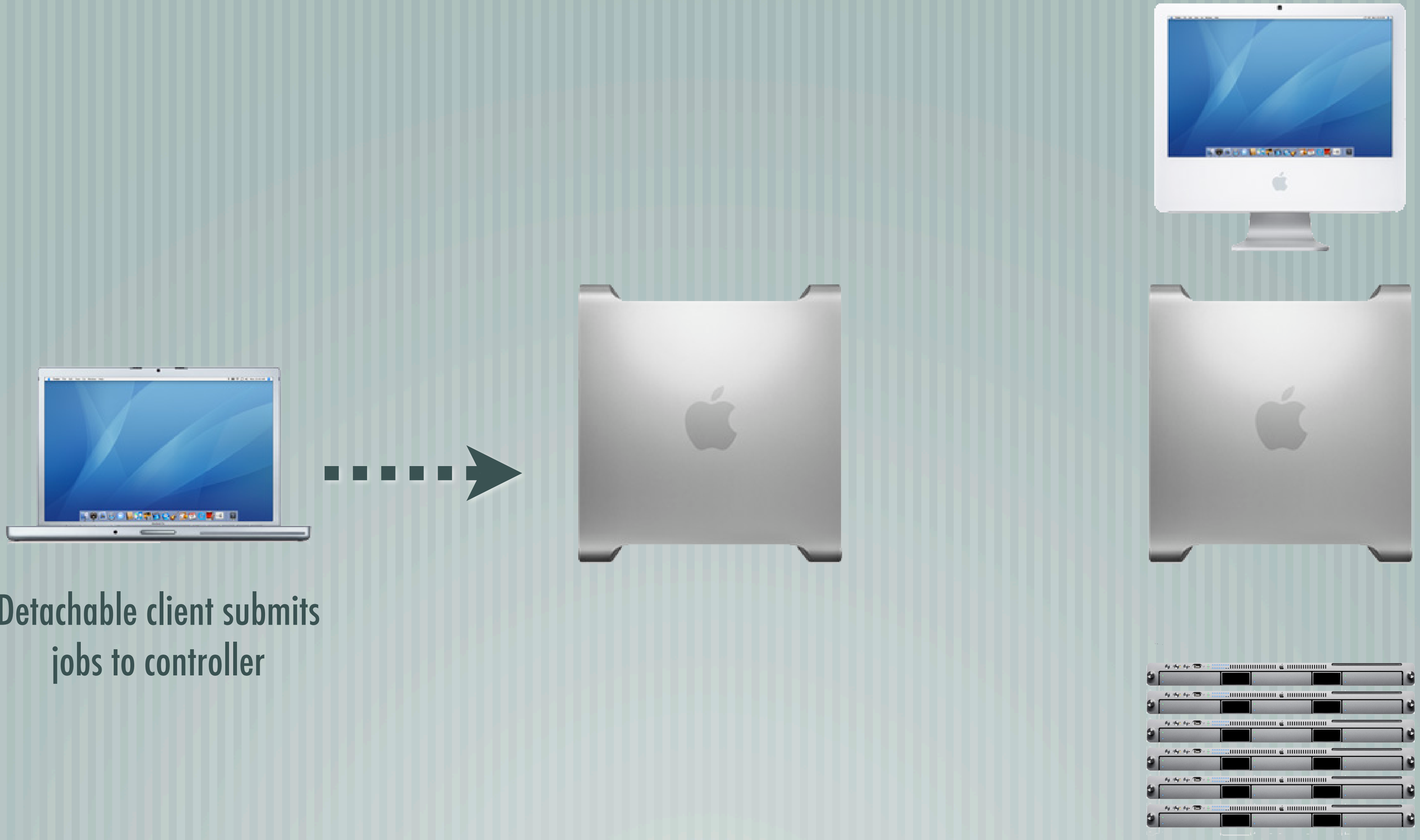
- Xgrid overview
- MIT cluster implementation
- STAR Unified Meta-Scheduler support
- Globus Toolkit integration

Xgrid Overview

- Distributed computing architecture from Apple’s Advanced Computation Group
- Promise: “Distributed computing made simple”
- Version 1.0 built into every copy of OS X 10.4
  - simplifies configuration
  - unique scalability opportunities
- Compatible with dedicated cluster or ad-hoc resource harvesting workflows
- Deployments in biochemical modeling, genetic sequencing, video encoding among others

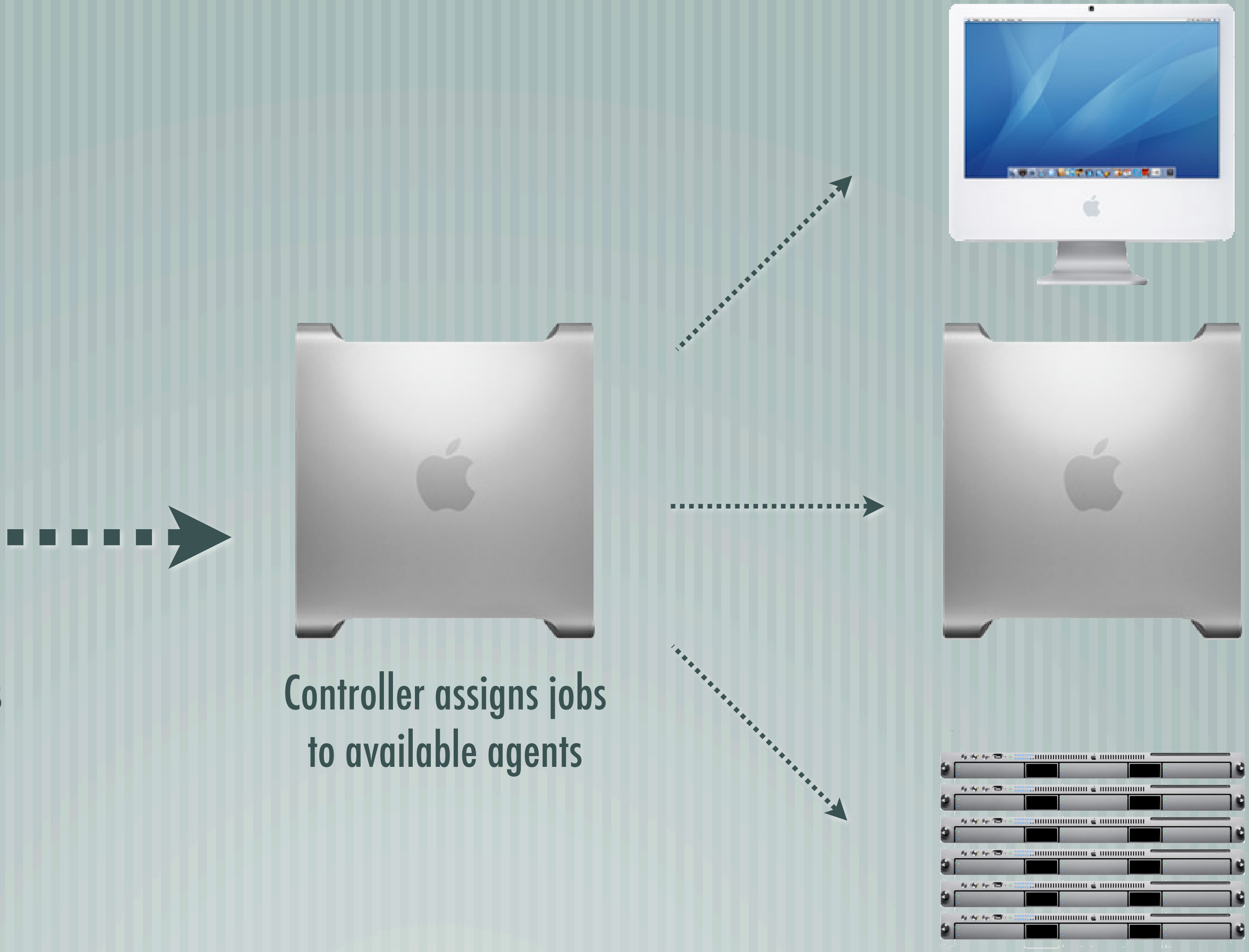
Terminology & Workflow

Terminology & Workflow



Detachable client submits jobs to controller

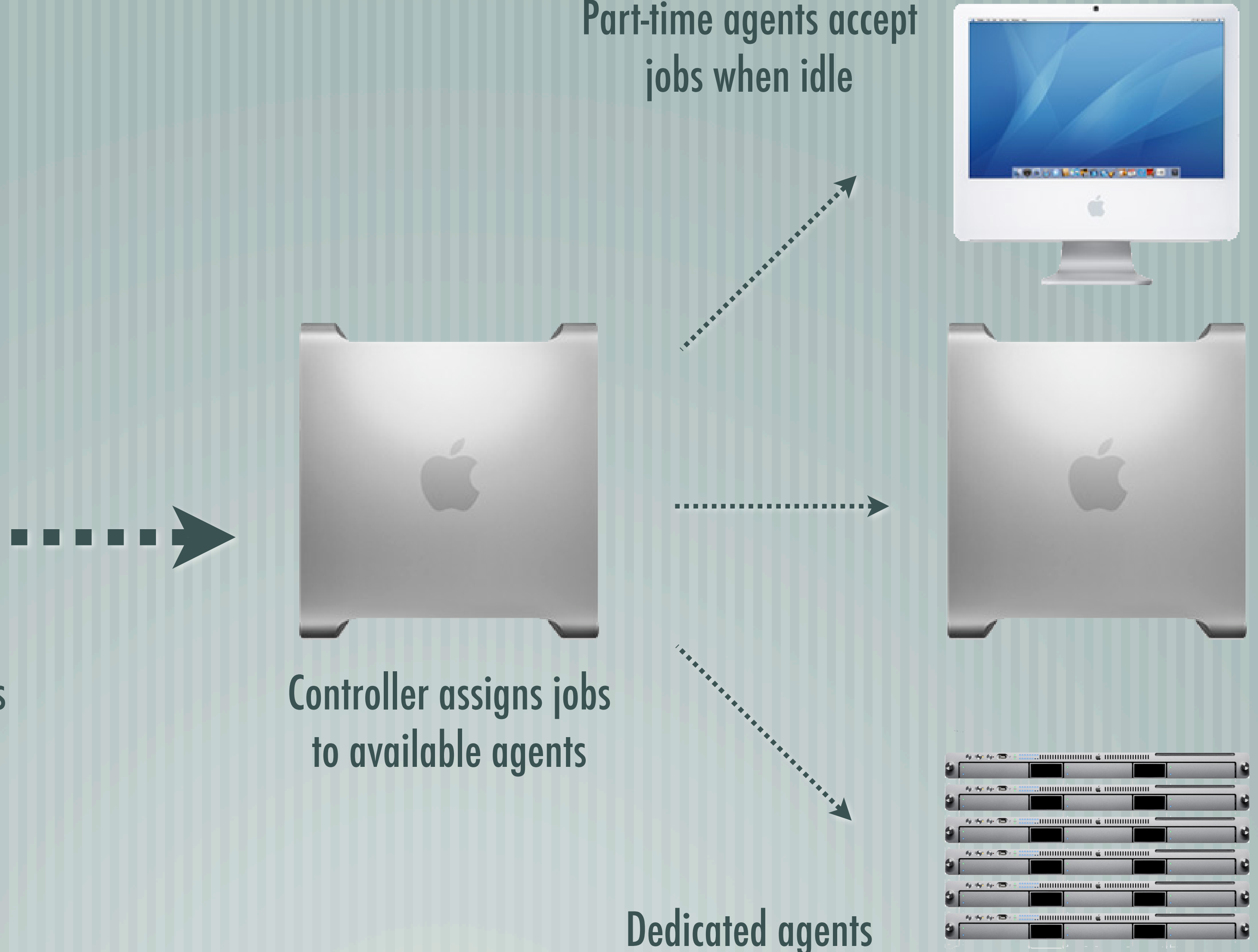
Terminology & Workflow



Detachable client submits jobs to controller

Controller assigns jobs to available agents

Terminology & Workflow



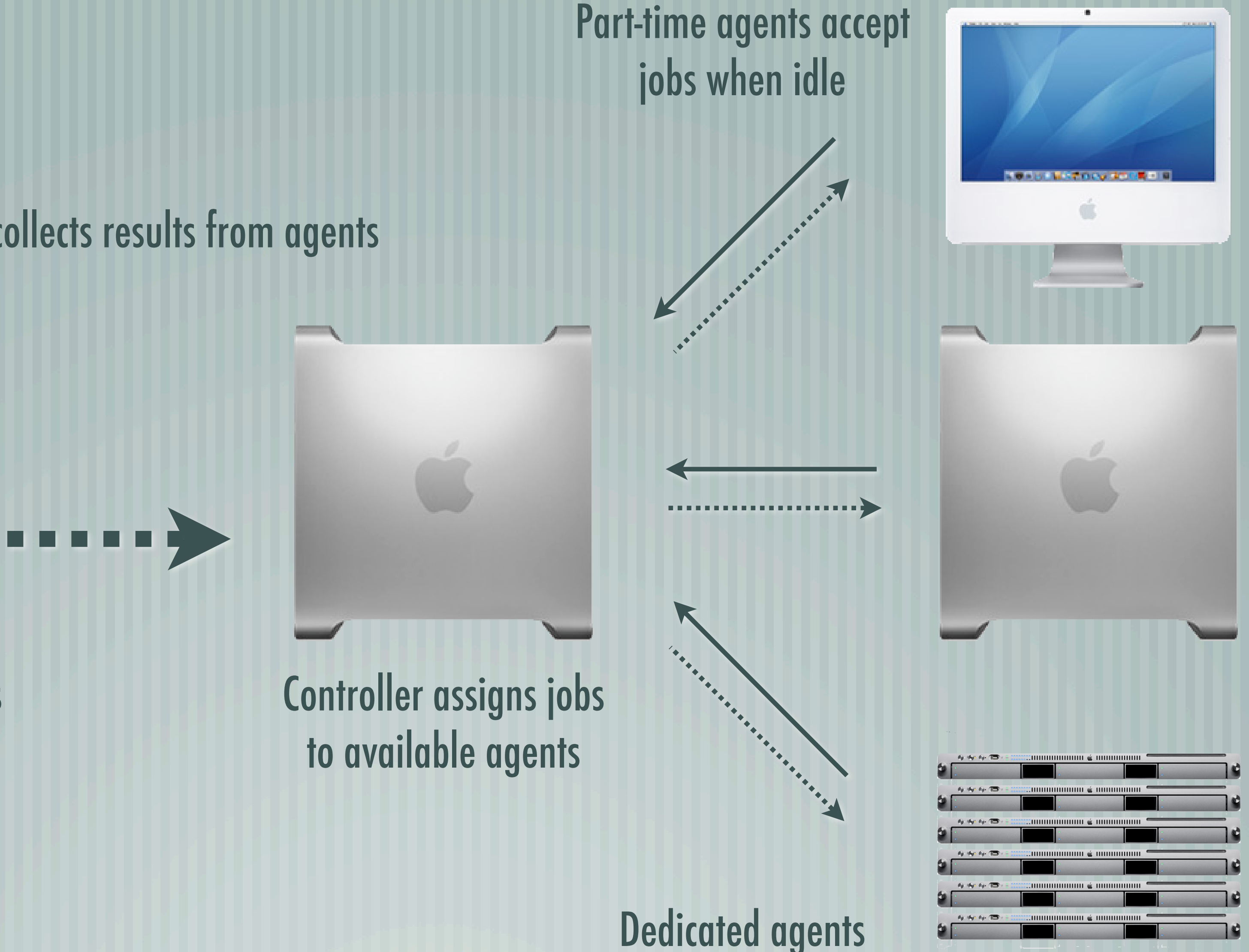
Part-time agents accept jobs when idle

Detachable client submits jobs to controller

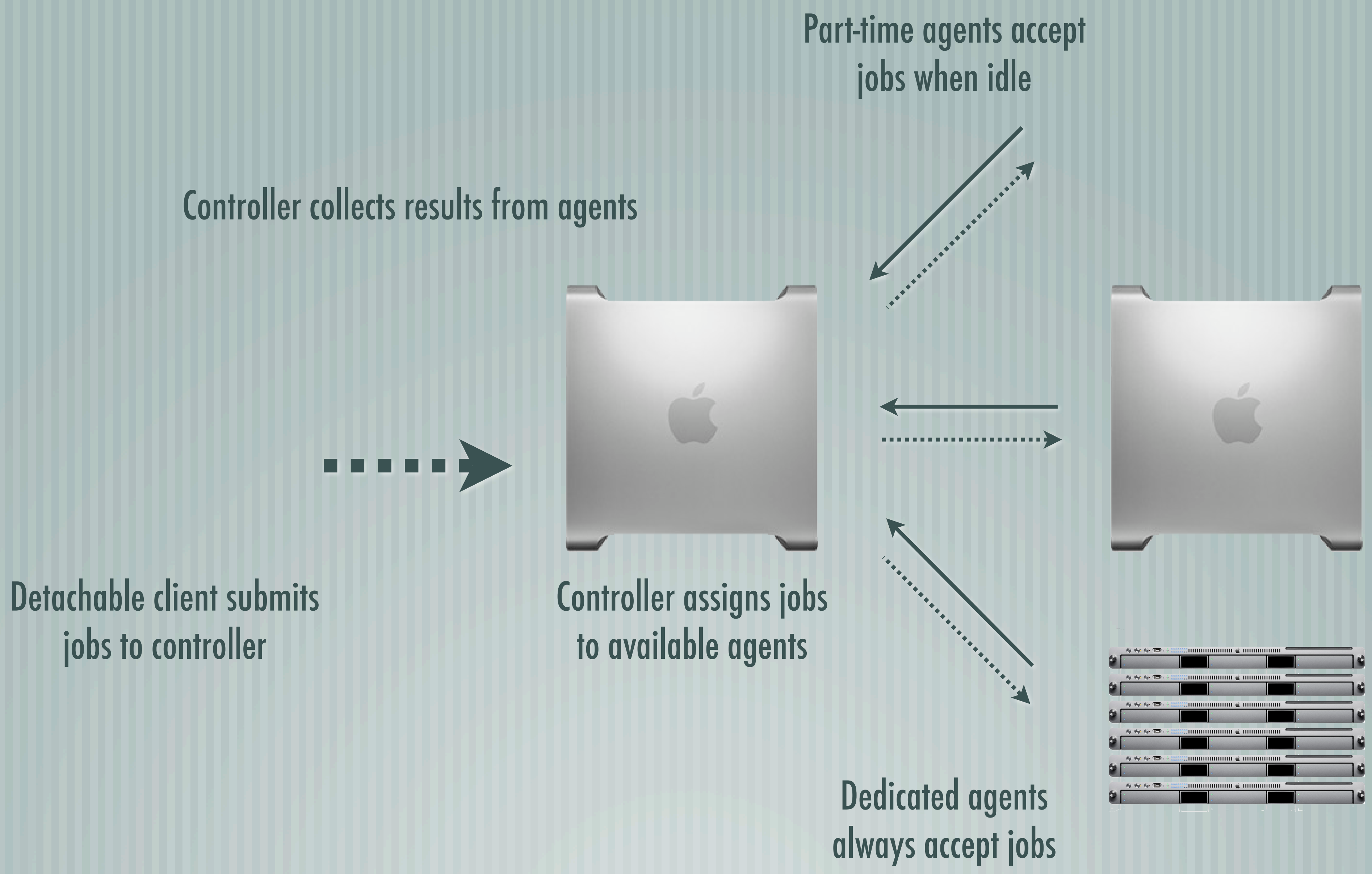
Controller assigns jobs to available agents

Dedicated agents always accept jobs

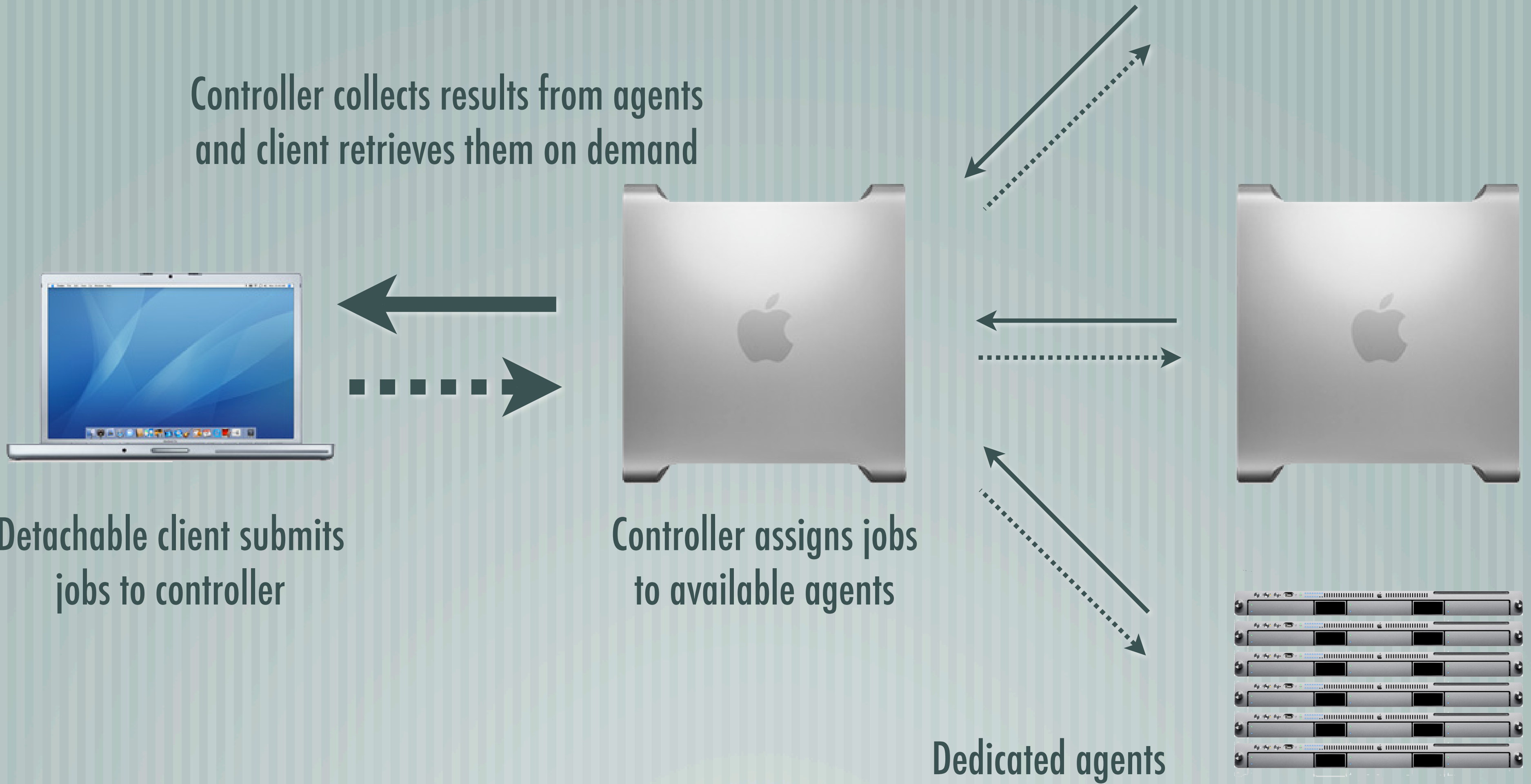
Terminology & Workflow


Part-time agents accept jobs when idle
Controller collects results from agents
Detachable client submits jobs to controller
Controller assigns jobs to available agents
Dedicated agents always accept jobs

Terminology & Workflow


Part-time agents accept jobs when idle
Controller collects results from agents
Detachable client submits jobs to controller
Controller assigns jobs to available agents
Dedicated agents always accept jobs

Terminology & Workflow


Part-time agents accept jobs when idle
Controller collects results from agents and client retrieves them on demand
Detachable client submits jobs to controller
Controller assigns jobs to available agents
Dedicated agents always accept jobs

Xgrid Technical Details

- All communications (even data transfer!) use a lightweight XML protocol built on BEEP
- No ports need to be opened on clients or agents; controller listens on 4111
- Optional Kerberized authentication available separately for clients and agents
  - if enabled on both, jobs execute as authenticated user (otherwise all run as nobody)
- MPI support with OpenMPI
- Job scheduling is just FIFO
- Accessible via command-line tool or native Cocoa API

An Xgrid cluster for HENP

- MIT site specifics
  - machines offer spare cycles
  - no centralized admin control
  - base OS X 10.4 assumed

} Minimal requirements on agents allow for easy scalability

- Admin access only needed when agents join for the first time
- HE(N)P libraries (ROOT, GEANT, CERNLIB, etc.) maintained on NFS for x86 and PPC – jobs load what they need at runtime
- Package management allows for easy installation of additional libraries as needed

Xgrid Data Transfer

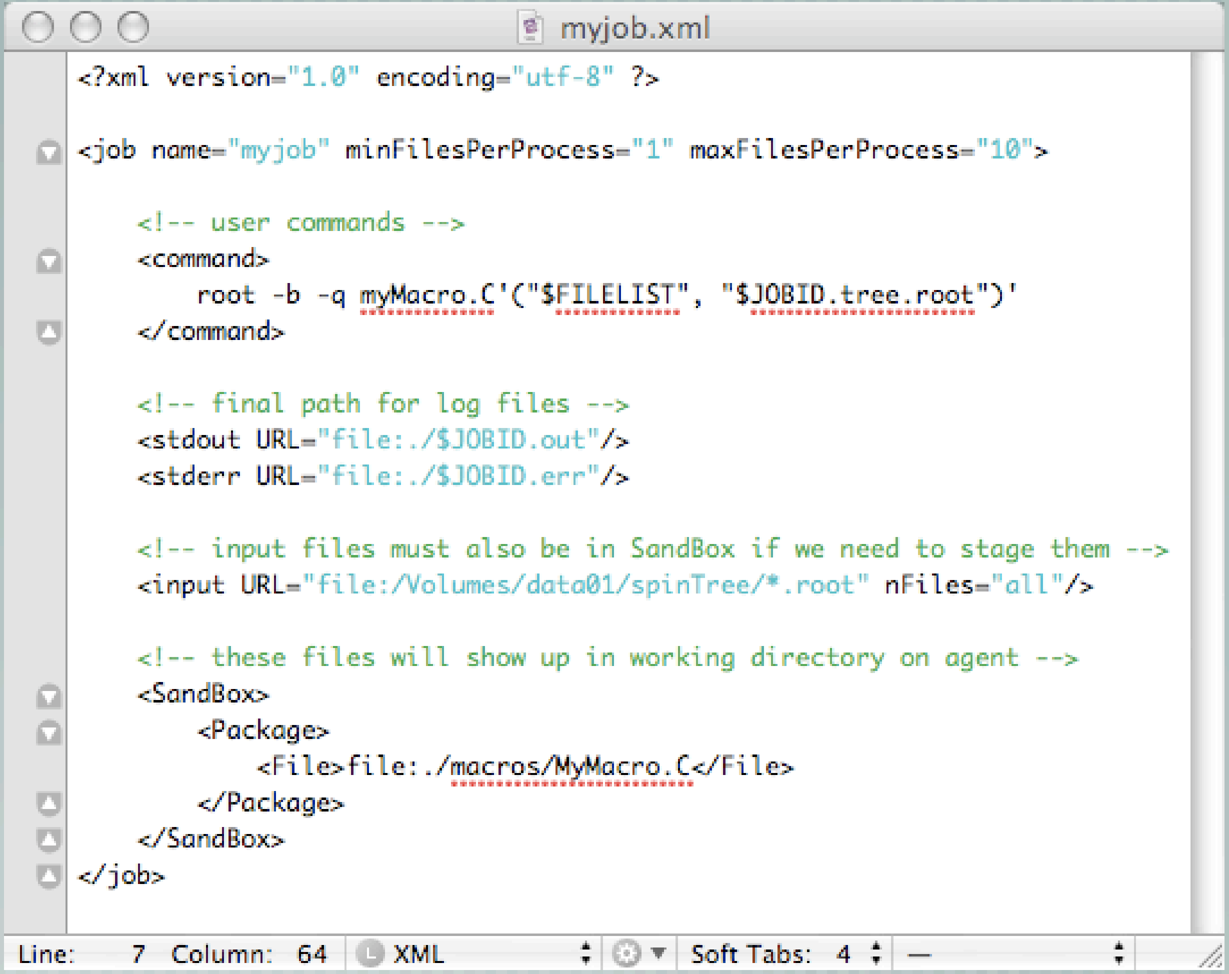
- I/O can be accomplished through Xgrid or using shared filesystem
- Xgrid protocol is flexible and secure, but performance can be an issue ...
  - Base64 encoding increases volume by 35%
  - All data must pass through controller DB → introduces extra overhead and potential bottleneck
  - #1 source of controller instabilities in 1.5 years of operation
- On the other hand, un-Kerberized jobs run as user nobody, so output directory on shared FS would be world-writeable
- No "one-size-fits-all" solution

STAR Unified Meta-Scheduler

- How do we really take advantage of this cluster we've built?
  - xgrid -job submit job1.sh
  - xgrid -job submit job2.sh, ...
- That won't scale well – need a tool to simplify processing of large datasets
- STAR Scheduler (SUMS) offers what we want
  - User gives high-level XML description of job; SUMS figures out how to get it done
  - Supports LSF, Condor, Condor-G, SGE, and now Xgrid
- **Business as usual for STAR users**

star-submit myjob.xml

- Scheduler goes to work:
  - Splits request into jobs
  - Generates shell scripts
  - Chooses queue
  - Submits jobs
  - Retrieves results
  - Cleans up queue



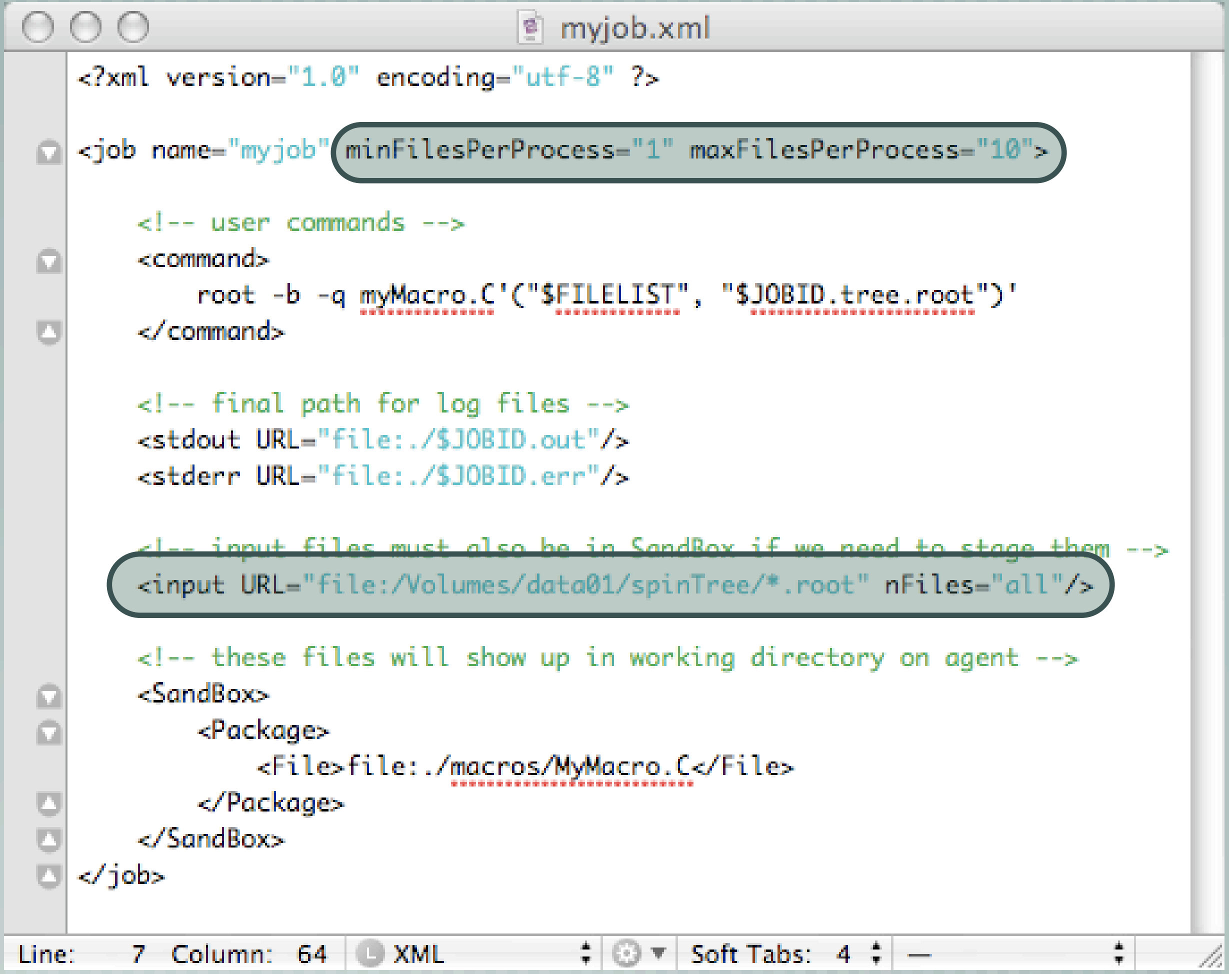
```
<?xml version="1.0" encoding="utf-8" ?>

<job name="myjob" minFilesPerProcess="1" maxFilesPerProcess="10">

    <!-- user commands -->
    <command>
        root -b -q myMacro.C'("$FILELIST", "$JOBID.tree.root")'
    </command>

    <!-- final path for log files -->
    <stdout URL="file:./$JOBID.out"/>
    <stderr URL="file:./$JOBID.err"/>

    <!-- input files must also be in SandBox if we need to stage them -->
    <input URL="file:/Volumes/data01/spinTree/*.root" nFiles="all"/>

    <!-- these files will show up in working directory on agent -->
    <SandBox>
        <Package>
            <File>file:./macros/MyMacro.C</File>
        </Package>
    </SandBox>
</job>
```

Line: 7 Column: 64 XML Soft Tabs: 4

CHEP 04 - J. Lauret, “SUMS: A front end around evolving technologies for user analysis and data production”

star-submit myjob.xml

- Scheduler goes to work:
  - Splits request into jobs
  - Generates shell scripts
  - Chooses queue
  - Submits jobs
  - Retrieves results
  - Cleans up queue



myjob.xml

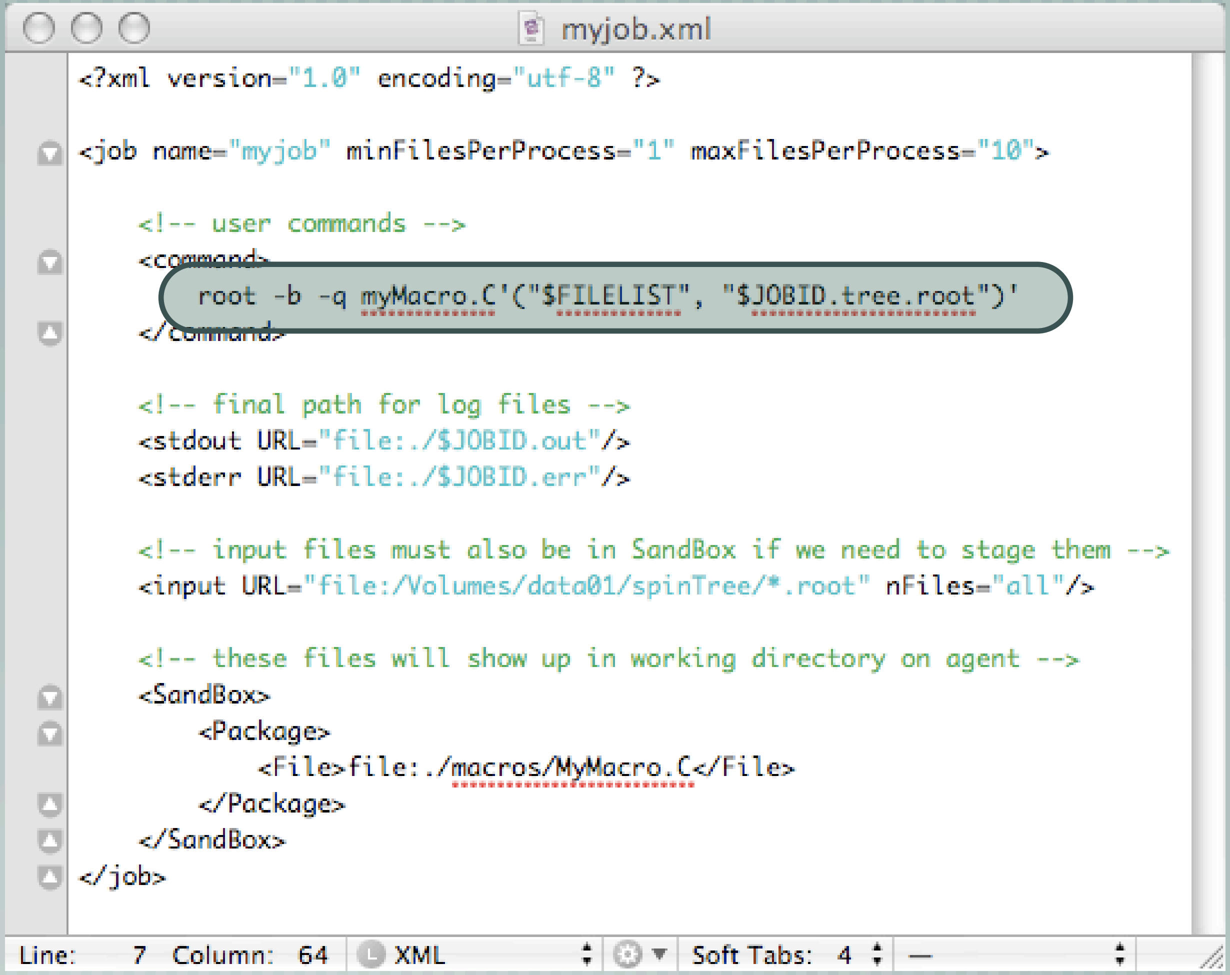
```
<?xml version="1.0" encoding="utf-8" ?>

<job name="myjob" minFilesPerProcess="1" maxFilesPerProcess="10">

    <!-- user commands -->
    <command>
        root -b -q myMacro.C'("$FILELIST", "$JOBID.tree.root")'
    </command>

    <!-- final path for log files -->
    <stdout URL="file:./$JOBID.out"/>
    <stderr URL="file:./$JOBID.err"/>

    <!-- input files must also be in SandBox if we need to stage them -->
    <input URL="file:/Volumes/data01/spinTree/*.root" nFiles="all"/>

    <!-- these files will show up in working directory on agent -->
    <SandBox>
        <Package>
            <File>file:./macros/MyMacro.C</File>
        </Package>
    </SandBox>
</job>
```

Line: 7 Column: 64 XML Soft Tabs: 4

CHEP 04 - J. Lauret, “SUMS: A front end around evolving technologies for user analysis and data production”

star-submit myjob.xml

- Scheduler goes to work:
  - Splits request into jobs
  - Generates shell scripts
  - Chooses queue
  - Submits jobs
  - Retrieves results
  - Cleans up queue



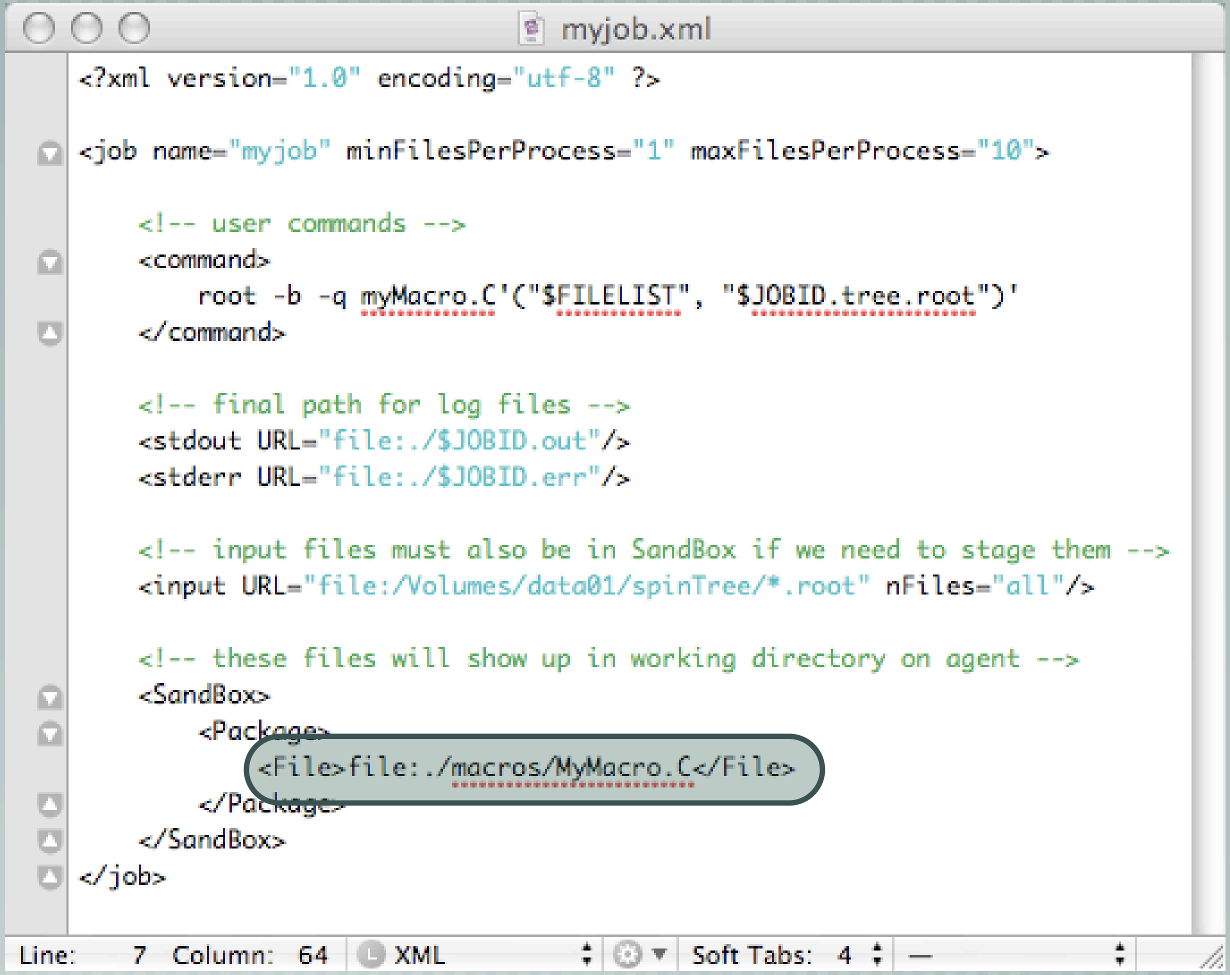
```xml
<?xml version="1.0" encoding="utf-8" ?>

<job name="myjob" minFilesPerProcess="1" maxFilesPerProcess="10">

    <!-- user commands -->
    <command>
        root -b -q myMacro.C'("$FILELIST", "$JOBID.tree.root")'
    </command>

    <!-- final path for log files -->
    <stdout URL="file:./$JOBID.out"/>
    <stderr URL="file:./$JOBID.err"/>

    <!-- input files must also be in SandBox if we need to stage them -->
    <input URL="file:/Volumes/data01/spinTree/*.root" nFiles="all"/>

    <!-- these files will show up in working directory on agent -->
    <SandBox>
        <Package>
            <File>file:./macros/MyMacro.C</File>
        </Package>
    </SandBox>
</job>
```

Line: 7 Column: 64 XML Soft Tabs: 4 —

CHEP 04 - J. Lauret, “SUMS: A front end around evolving technologies for user analysis and data production”

star-submit myjob.xml

- Scheduler goes to work:
  - Splits request into jobs
  - Generates shell scripts
  - Chooses queue
  - Submits jobs
  - Retrieves results
  - Cleans up queue



myjob.xml

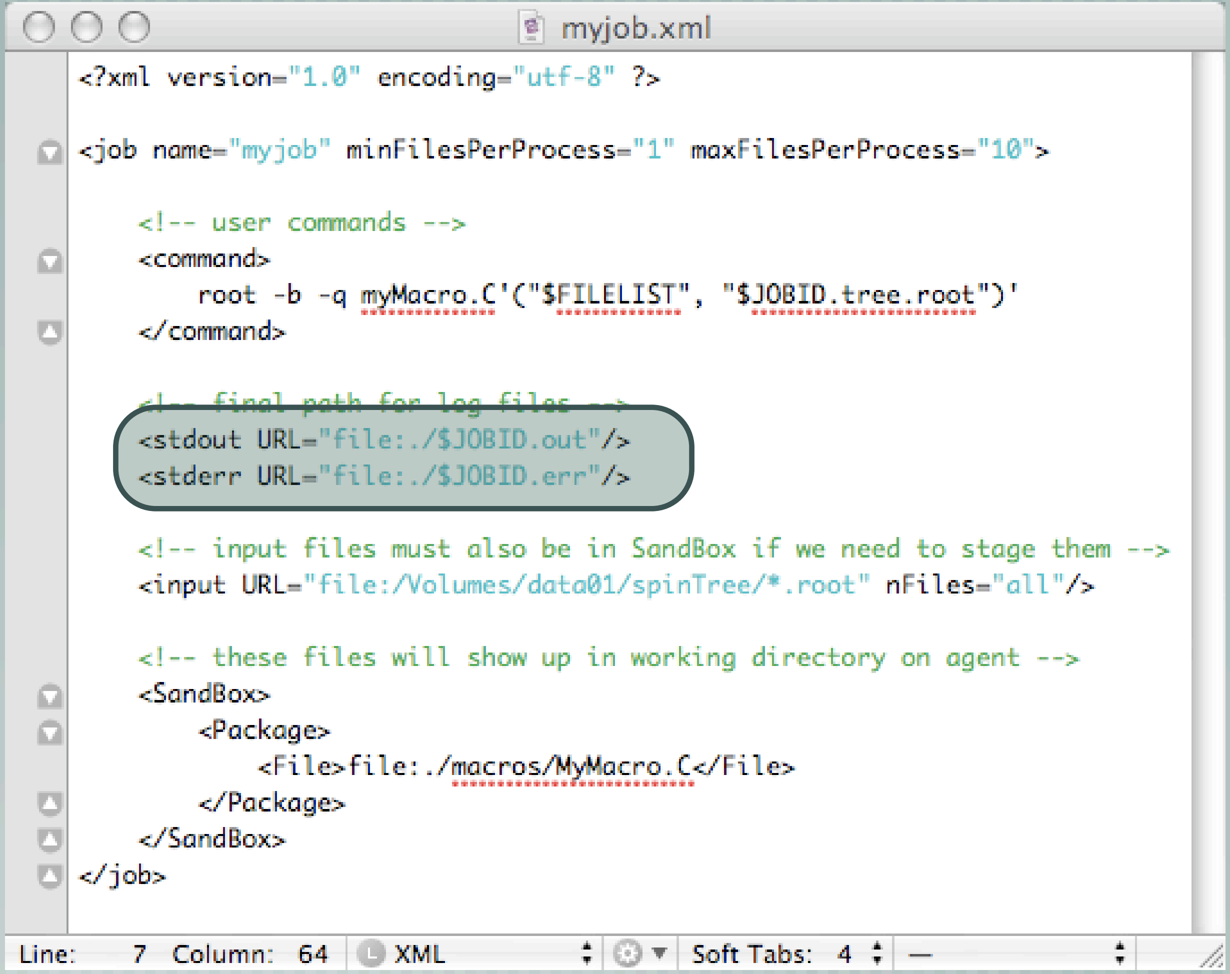
```
<?xml version="1.0" encoding="utf-8" ?>

<job name="myjob" minFilesPerProcess="1" maxFilesPerProcess="10">

    <!-- user commands -->
    <command>
        root -b -q myMacro.C'("$FILELIST", "$JOBID.tree.root")'
    </command>

    <!-- final path for log files -->
    <stdout URL="file:./$JOBID.out"/>
    <stderr URL="file:./$JOBID.err"/>

    <!-- input files must also be in SandBox if we need to stage them -->
    <input URL="file:/Volumes/data01/spinTree/*.root" nFiles="all"/>

    <!-- these files will show up in working directory on agent -->
    <SandBox>
        <Package>
            <File>file:./macros/MyMacro.C</File>
        </Package>
    </SandBox>
</job>
```

Line: 7 Column: 64 | XML | Soft Tabs: 4 | —

CHEP 04 - J. Lauret, “SUMS: A front end around evolving technologies for user analysis and data production”

star-submit myjob.xml

- Scheduler goes to work:
  - Splits request into jobs
  - Generates shell scripts
  - Chooses queue
  - Submits jobs
  - Retrieves results
  - Cleans up queue


myjob.xml

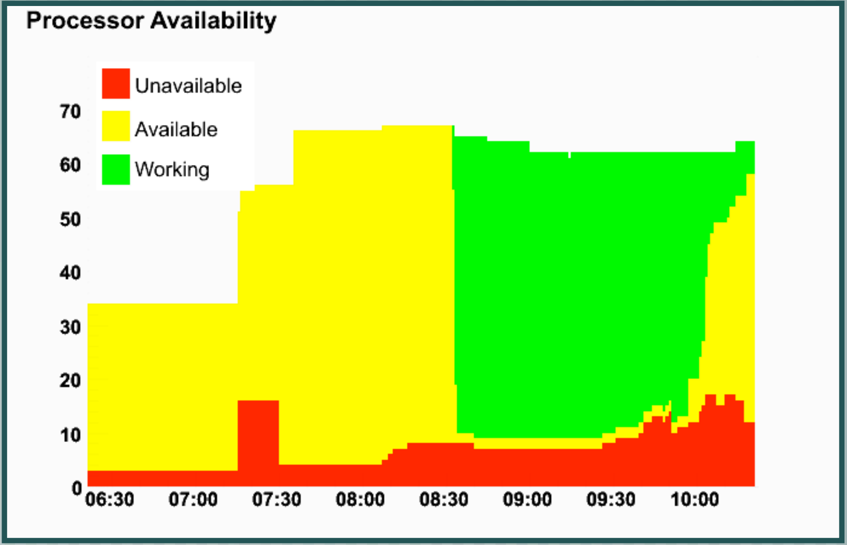
```xml
<?xml version="1.0" encoding="utf-8" ?>

<job name="myjob" minFilesPerProcess="1" maxFilesPerProcess="10">

    <!-- user commands -->
    <command>
        root -b -q myMacro.C'("$FILELIST", "$JOBID.tree.root")'
    </command>

    <!-- final path for log files -->
    <stdout URL="file:./$JOBID.out"/>
    <stderr URL="file:./$JOBID.err"/>

    <!-- input files must also be in SandBox if we need to stage them -->
    <input URL="file:/Volumes/data01/spinTree/*.root" nFiles="all"/>

    <!-- these files will show up in working directory on agent -->
    <SandBox>
        <Package>
            <File>file:./macros/MyMacro.C</File>
        </Package>
    </SandBox>
</job>
```

Line: 7 Column: 64 XML Soft Tabs: 4

CHEP 04 - J. Lauret, “SUMS: A front end around evolving technologies for user analysis and data production”

Current Status

- Stable running since March 2006
- 50 machines from around MIT
- 20,000 jobs and counting
- A unique campus grid



Project initiated to aid STAR simulations and user analyses, but has since proven useful to multiple research groups in MIT's Laboratory for Nuclear Science

Towards GRID Integration

- Local Resource Manager support in GT GRAM requires:
  - Perl scheduler adapter to submit and cancel jobs
  - C scheduler event generator to monitor log files for jobs state changes
- Preliminary versions of both codes are being tested
  - special steps required to export data from controller DB after job finishes
  - generation of Kerberos credentials from GSI certificates still in development
- GRAM integration complements future OSG VDT server support for OS X

Towards GRID Integration

- Local Resource Manager support in GT GRAM requires:
  - Perl scheduler adapter to submit and cancel jobs
  - C scheduler event generator to monitor log files for jobs state changes
- Preliminary versions of both codes are being tested
  - special steps required to export data from controller DB after job finishes
  - generation of Kerberos credentials from GSI certificates still in development
- GRAM integration complements future OSG VDT server support for OS X

Beta exists

Summary & Outlook

- Unique application of Xgrid technology in HENP computing
- Grassroots project providing immediately useful resource to MIT's Laboratory for Nuclear Science
- Minimal requirements on agents promise scalability and a clear upgrade path
- SUMS support for Xgrid ensures accessibility for STAR users
- Future integration into GRAM and the OSG VDT will allow an Xgrid cluster to be a full partner in grid computing initiatives.