Achieving production-level use of HEP software at the Argonne Leadership Computing Facility

Thomas D. Uram  
Michael E. Papka  
Argonne Leadership Computing Facility,  
Argonne National Laboratory

J. Taylor Childers  
Thomas J. LeCompte  
High Energy Physics Division,  
Argonne National Laboratory

Doug Benjamin  
Duke University
Mira - Leadership-class Supercomputer at Argonne
Why use Mira for HEP?

- The HEP community is a leader in deploying grid computing
  - ~3 billion core hours delivered by grid each year
- Run II will demand a significant increase in computing resources
- Supercomputers will be an essential platform
  - to expand the computing resources available
  - to absorb spikes in computing demand
  - to produce events for complex processes not currently possible on the grid
- We are adapting legacy HEP codes to run on today’s (and tomorrow’s) supercomputers
Applications running on Mira

- Over the past two years, we have adapted several HEP codes to run on Mira
  - Alpgen
  - Sherpa
  - Pythia
  - MCFM
  - Geant4

- But which codes can we run in parallel on Mira?
  - Event generation codes are a good choice
- How to integrate with PanDA?
Adapting Alpgen to run on Mira

- Mostly straightforward compilation
- Use MPI ranks to determine random seeds
- Use MPI to distribute input data (configuration + grids + pdf)
- Decomposed execution into serial and parallel portions
Adapting Alpgen to run on Mira

- Mostly straightforward compilation
- Use MPI ranks to determine random seeds
- Use MPI to distribute input data (configuration + grids + pdf)
- Decomposed execution into serial and parallel portions
- Coupled event generation and unweighting in single Mira job
Adapting Alpgen to run on Mira

- Mostly straightforward compilation
- Use MPI ranks to determine random seeds
- Use MPI to distribute input data (configuration + grids + pdf)
- Decomposed execution into serial and parallel portions
- Combined event generation + unweighting + aggregation
Adapting Alpgen to run on Mira

- Mostly straightforward compilation
- Use MPI ranks to determine random seeds
- Use MPI to distribute input data (configuration + grids + pdf)
- Decomposed execution into serial and parallel portions
- Combined event generation + unweighting + aggregation
Adapting Alpgen to run on Mira

- Mostly straightforward compilation
- Use MPI ranks to determine random seeds
- Use MPI to distribute input data (configuration + grids + pdf)
- Decomposed execution into serial and parallel portions

For more details on adapting Alpgen for Mira, see Taylor Childers’ CHEP2015 talk "Simulation of LHC events on a million threads (#536)"
How to integrate with PanDA?

1. Integration
2. Event generation
3. Unweighting
4. MPI-IO aggregation

Mira

x86 cluster

PanDA
How to integrate with PanDA?

- A job management service, Argo, runs within ANL–HEP domain to control job execution
  - Stores Athena job description
  - Submits integration job to x86 cluster
  - Submits event generation + unweighting job to Mira
  - Converts to pool files
  - Sends files to the grid

- Python/Django–based, backed with MySQL (or other databases)
How to integrate with PanDA?

- A job execution service, Balsam, prepares and runs jobs on target resources
  - Subscribes to job message queue
  - Stages in job input
  - Submits job to schedulers (Condor, Cobalt, Torque)
  - Monitors running job
  - Stages job output to specified destination
  - Sends completion message to message queue
- Python/Django-based, backed with MySQL (or other databases)
Integration with PanDA

PanDA

Argo

message queues

Balsam

integration

x86 cluster

1

Balsam

event generation

2

compute node ramdisk

Mira

Balsam

unweighting

3

MPI-IO aggregation

4

Thomas D. Uram, Argonne Leadership Computing Facility
Computing in High Energy Physics 2015
Progress using 50M hour ALCC award

HadronSim
Machine: MIRA
Allocation: 50,000,000
Usage: 34,234,852.67 (68.5%)
2014-07-01 to 2015-04-12

On track to use full allocation

8K+ nodes
16K+ nodes
<8K nodes
Next in Queue

- Working with PanDA team to build pilot job runner
- Integrate Sherpa
  - Sherpa has long been running on Mira, and has run through Argo
  - Scales to ~512 nodes; running as 256-node subjobs within larger jobs
  - Working (with Stefan Hoeche) to scale event generation to larger sizes
- Consider integrating Pythia
  - Pythia has long been running on Mira, and has run through Argo
  - Large memory footprint (2GB) initially restricted execution to 4 cores per node
  - Use LHAPDF6 to reduce memory footprint to 200MB: 64 cores per node now possible (LHAPDF5 reserves space for large uninitialized memory that bloats executable size)
Mira - Allocation Process

- How do projects get allocations on Mira?
  - Through a competitive proposal process operated by the U.S. Department of Energy
  - Open to projects worldwide
  - 5B hours awarded at Argonne in 2015
- INCITE program: 60%
  - Granted at DOE level
  - Largest, most strategic and most mature projects
- ASCR Leadership Computing Challenge (ALCC) program: 30%
  - Granted at DOE level
  - For projects vectored toward INCITE proposals
- Director’s Discretionary: 10%
  - Small startup allocations granted by ALCF director
Summary

- We are running Alpgen on the 49,152 nodes on Mira, the 10PFLOPs supercomputer at Argonne
- Using Argo and Balsam, we have used 34 million core hours on Mira (16 million more hours remaining this year)
  - Produced tens of millions of Alpgen events
  - Producing high multiplicity events (w/z+5, w/z+6) that can't be generated on the grid today
  - ANL/HEP is now the primary Alpgen event generation site for ATLAS
Acknowledgements

- This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.