Computational workflow of the LZ experiment

Venkitesh Ayyar,
Computation Research Divsion,
Lawrence Berkeley National Laboratory

CHEP 2019, Adelaide,
November 4-8, 2019

Joint work with W. Bhimji, M.E. Monzani, A. Naylor, S. Patton, Q. Riffard, C. Tull

For the LUX-ZEPLIN collaboration

Dark matter

Evidence of matter beyond SM:

- ★ Galaxy rotation curves
- ★ Cosmic Microwave Background
- ★ Gravitational lensing

Galaxy rotation curves

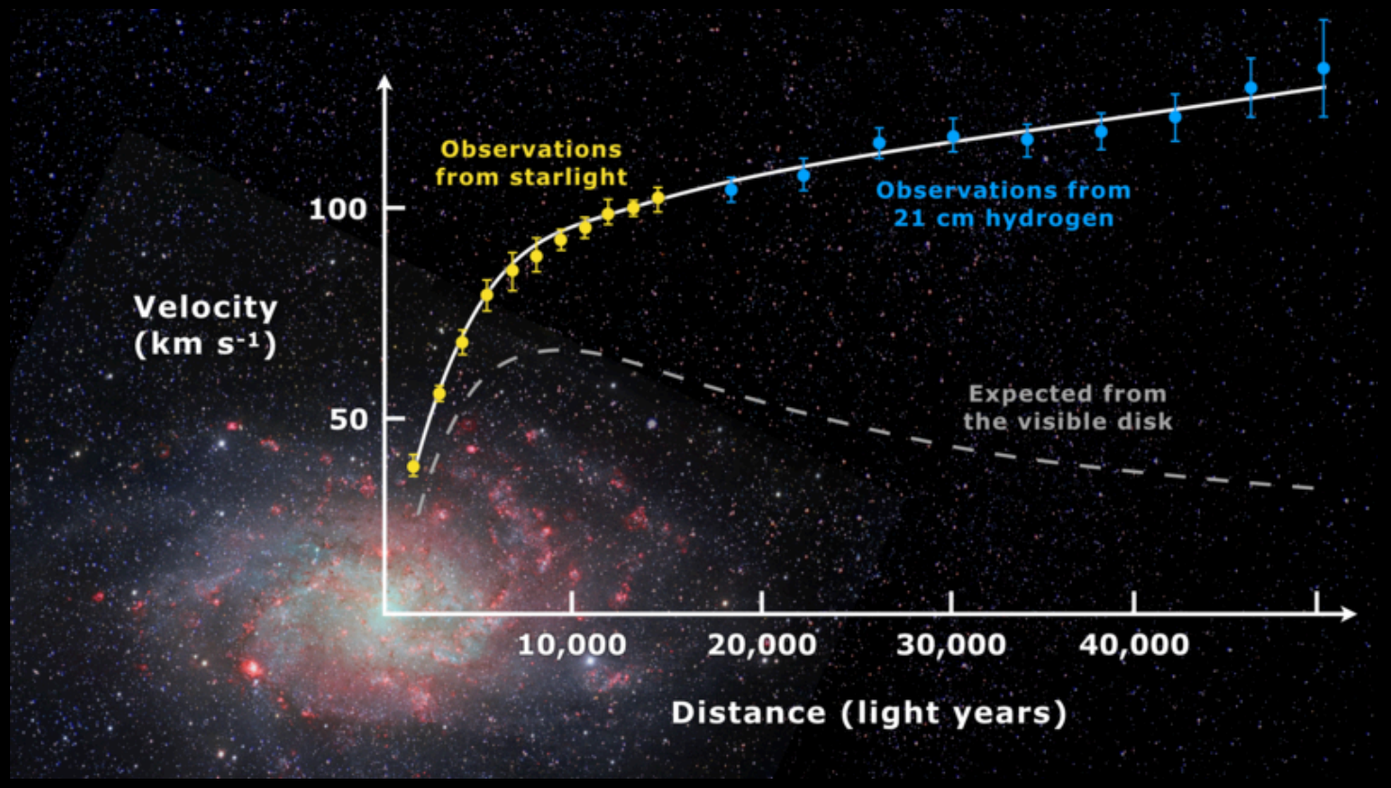


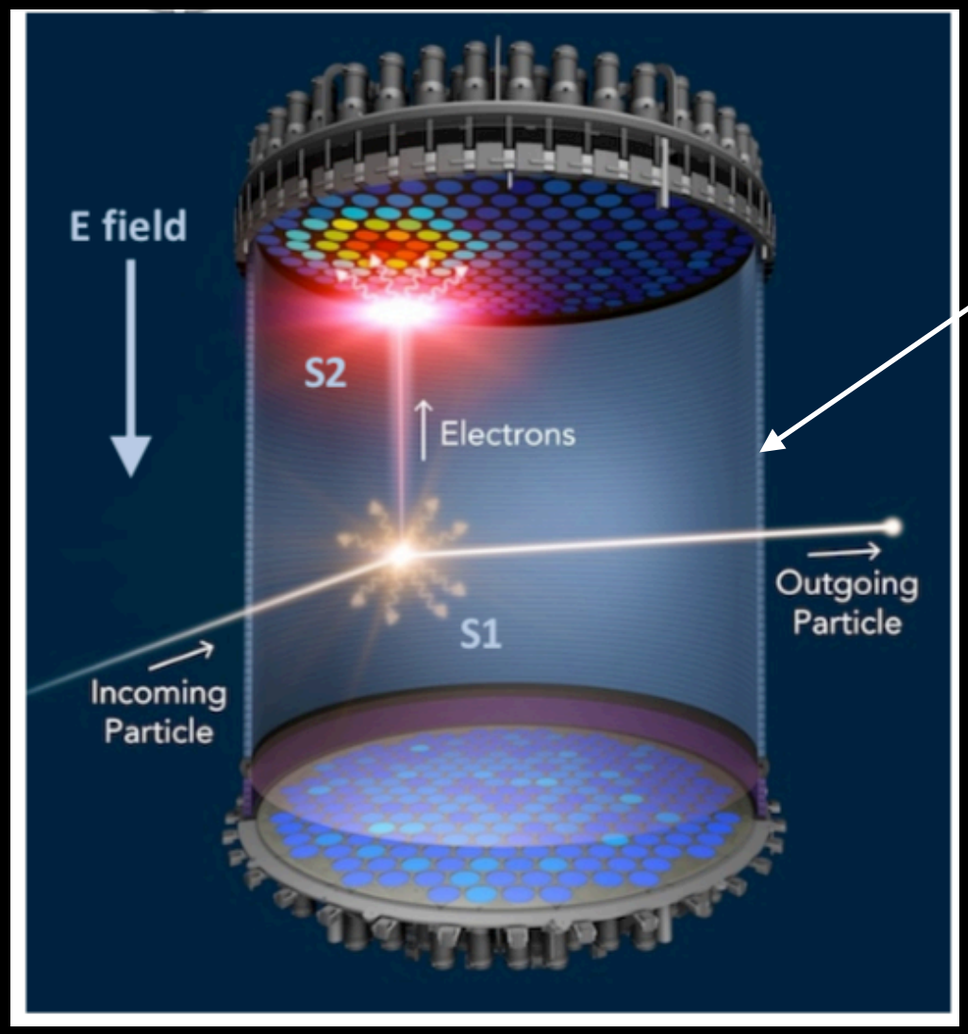
Figure from Wikipedia

Corbelli, E.; Salucci, P. (2000)
Rubin, V.; Thonnard, N.; Ford, W. K. Jr. (1980)

Estimated to constitute up to 85% of all matter
Different types of dark matter proposed
One of them is WIMP (Weakly Interacting Massive Particle)

Some current experiments: LZ, XENONnT, PandaX-4T

LZ experiment LUX + ZEPLIN



Tank containing 7 tonnes liquid Xe

- Direct detection experiment
- Looking for Weakly Interacting Massive Particles (WIMPS)

Detect scintillations in liquid Xe

Sanford Underground Research Facility, South Dakota

LZ collaboration

Collaboration of 37 institutes :
US, UK, Portugal, Russia, Korea

Data centers in both UK and US

Expected timelines:

Experimental installation: 2019
Data collection start: 2020

Data Challenges

MDC2: September 2018
MDC3 : October 2019

The Code

in C++

Time consuming part

Input

Detector configuration,

Runs much faster

BACCARAT

Simulates events using Geant4

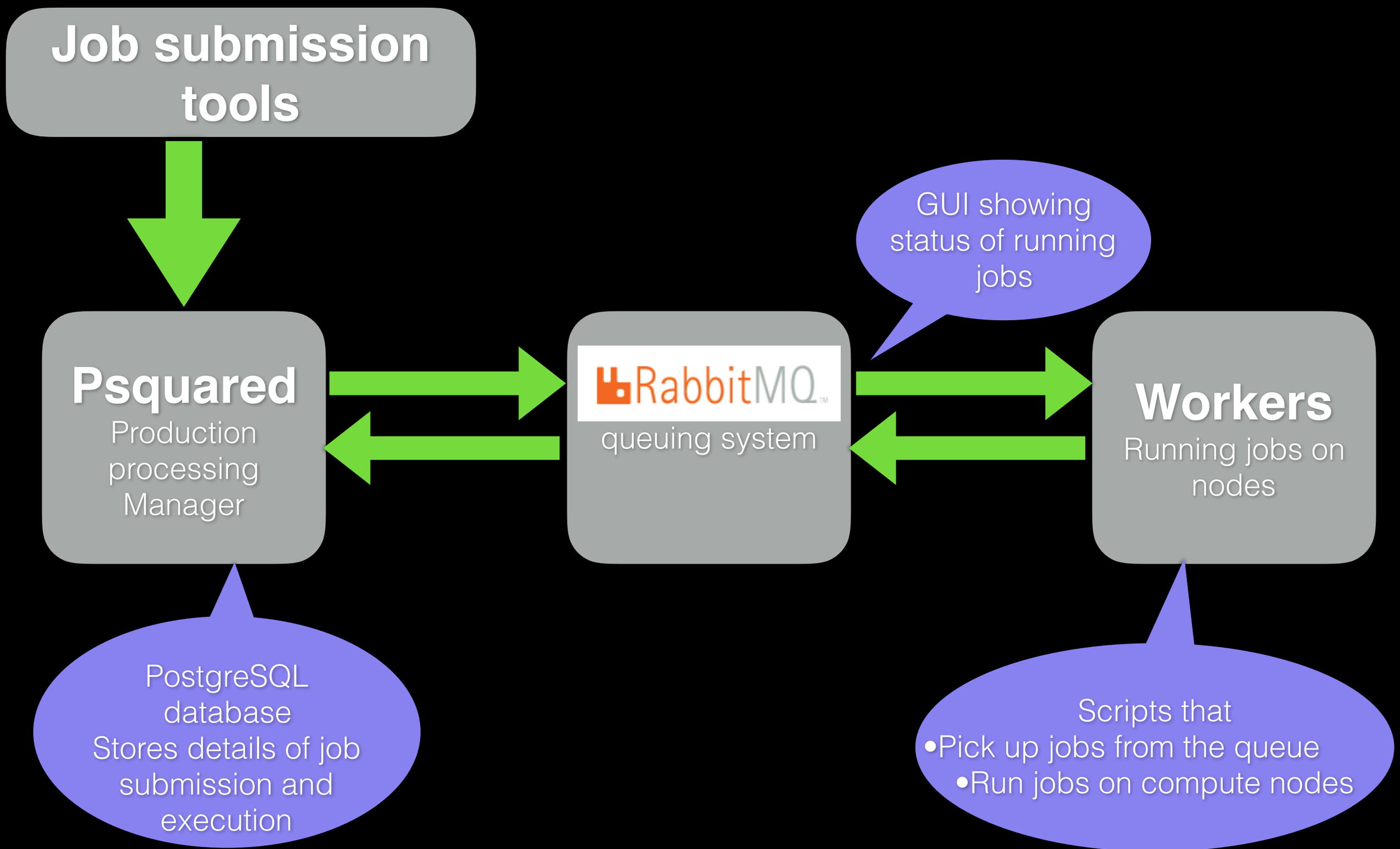
DER

Creates sensor data for the given events

LZap

Event reconstruction (Gaudi)

Computing workflow



Job submission

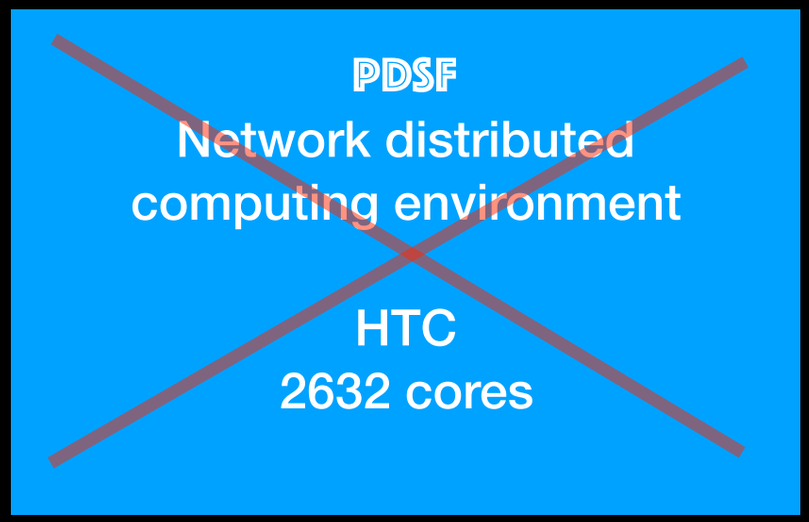
CLI
Command Line
Interface

- Python scripts to submit jobs
- Tools for debugging failed jobs for re-submission
- Interacts with Postgres Database via restful-interface
- Intended for use by production team

Web-based
Graphical User
Interface

- Job Submission Interface developed by UKDC
- Developed for USDC by Andrew Naylor
- Users can request to submit jobs
- Admin approval for submission

Computing resources at US Data Center (NERSC)



HPC

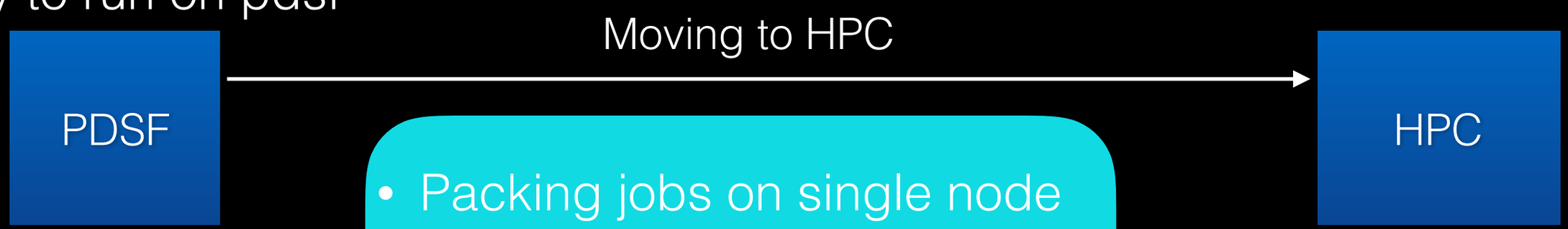
Machine	Details	Total nodes	Cores per node	Memory per node (GB)
~~Edison~~	Cray XC30	5586	24	64
Cori-Haswell	Cray XC40	2388	32	128
Cori- Knl	Cray XC40	9688	68	96

PDSF and Edison decommissioned
Currently only Cori is running
New machine Perlmutter is also HPC, with GPUs

→ LZ workflow had to be ported to HPC

Jobs are completely independent → Easy to parallelize

Easy to run on pdsf



- Packing jobs on single node
- End after longest job

Reasons to focus on HPC:

- Large investment by DOE in HPC exascale machines
- NERSC focus on large HPC machines
- Memory shared across cores: less chance of memory overload

Improving efficiency

Before September 2018

Avg mem usage ~ 4GB
Jobs per node = 32

High memory usage

- Restricts number jobs per node
- Increases job-crash incidents

Use of profiling and monitoring tools to fix memory leaks

Process monitor

Developed by ATLAS

https://github.com/HSF/prmon

Gives memory usage and other I/O data as a function of time

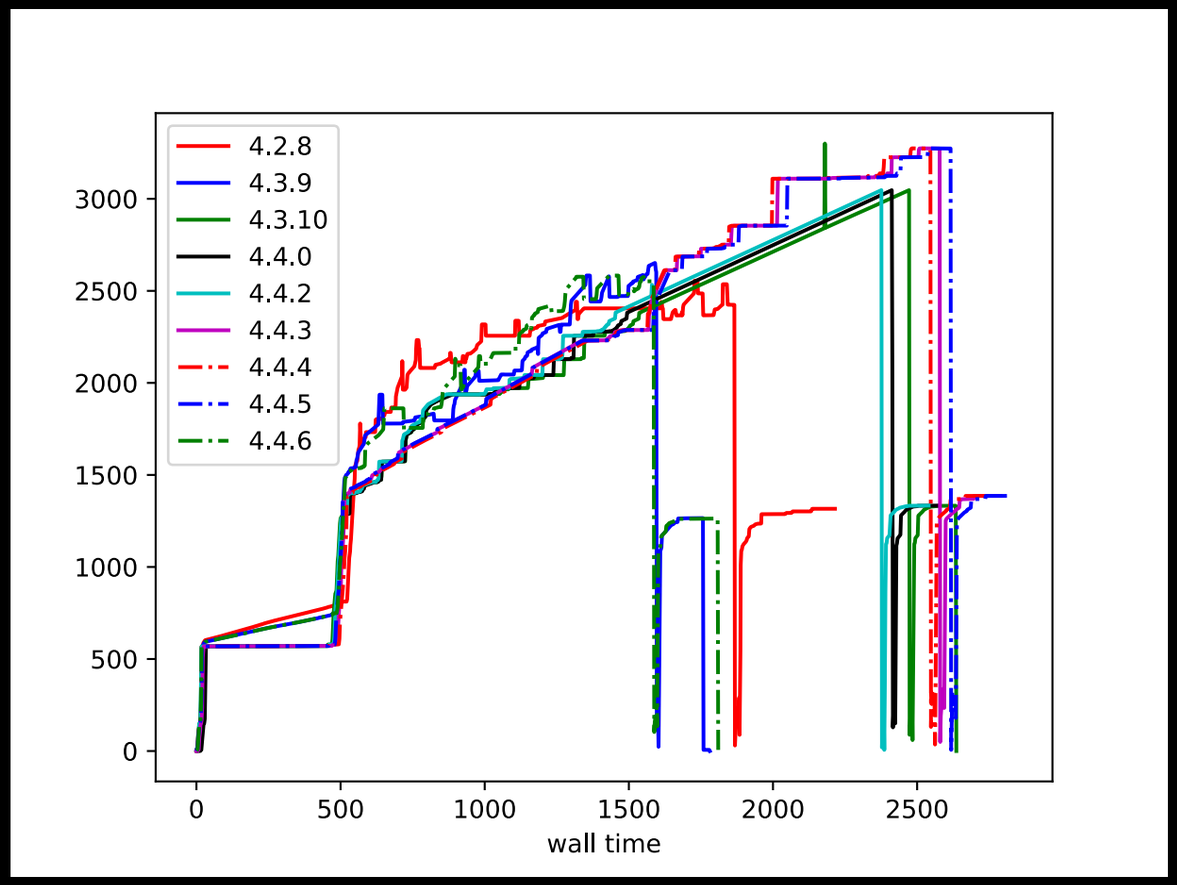
- Simple, easy to use
- Gives time map for a lot of quantities
- Lightweight
- Tacks on to pid, gives usage of all spawned processes

Quantity monitored	Description
vmem	Virtual Memory
pss	Proportional Set Size
rss	Resident Set Size
swap	Swap Size
utime	User CPU-time
stime	System CPU-time
wtime	Wall-time
rchar	I/O Read (rchar)
wchar	I/O Written (wchar)
read_bytes	I/O Read (read_bytes)
write_bytes	I/O Written (write_bytes)
rx_packets	Network Received (packets)
tx_packets	Network Transmitted (packets)
rx_bytes	Network Received (bytes)
tx_bytes	Network Transmitted (bytes)
nprocs	Number of Processes
nthreads	Number of Threads

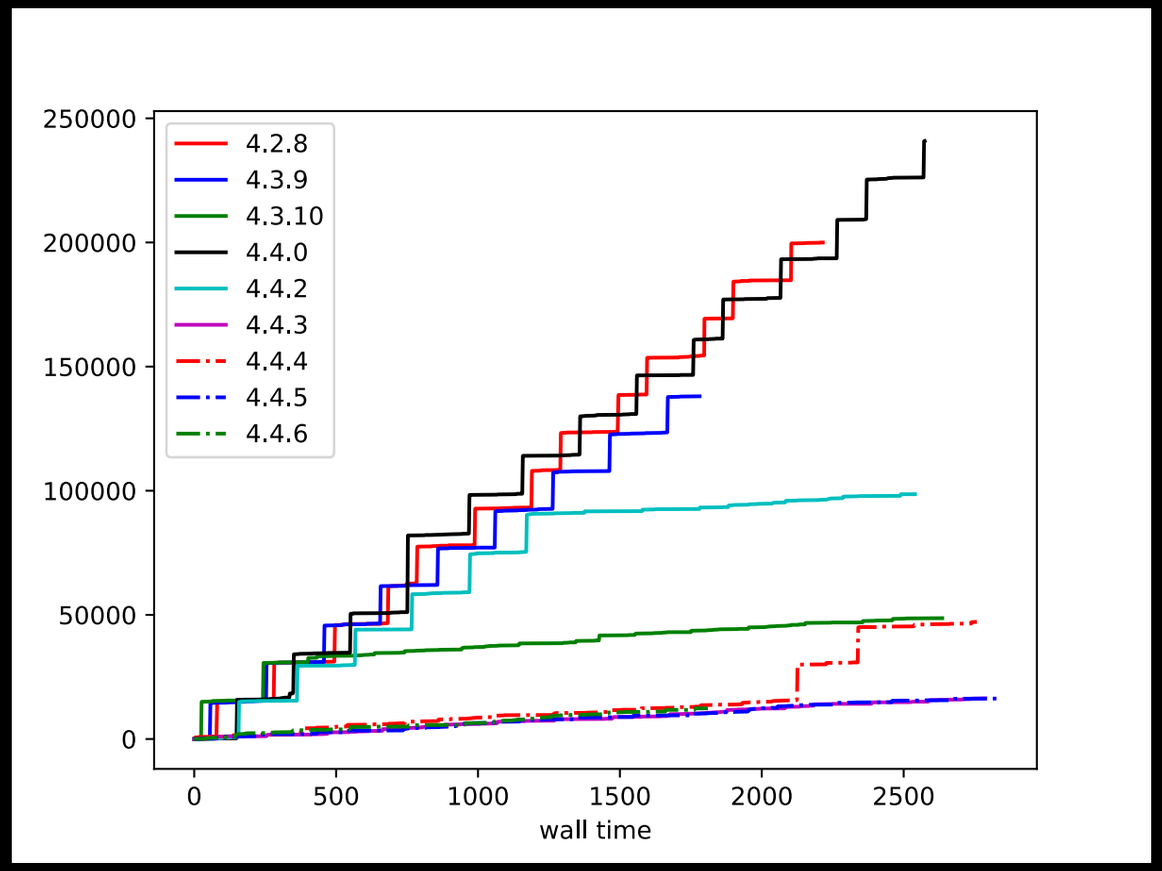
```
#### Run the MCprod script
shifter $script_loc/./RunMCprod.sh OUTPUT=$output_loc BACCARAT_VERSION=$bac_vers BACCARAT_MAC=$mac_loc BACCARAT_CODE=$bac_code ITEM=$ITEM &
####################
#### Link the pid to the job above
$prmon_loc --pid $! --filename $prmon_output --interval $interval
```

Further info

Memory usage in MB



Network received bytes



Has its own plotting code
Also supports GNU prof profiling or gperftools profiling

VTune

Developed by Intel

Tool for detailed profiling

- Collects a lot of data
- Runs a lot longer
- Requires full node

Analysis Target | Analysis Type | Collection Log | Summary | Bottom-up

Elapsed Time: 5492.083s

Allocation Size:	51 GB
Deallocation Size:	49 GB
Allocations:	232,619,626
Total Thread Count:	3
Paused Time:	0s

Top Memory-Consuming Functions

This section lists the most memory-consuming functions in your application.

Function	Memory Consumption	Allocation/Deallocation Delta	Allocations	Module
_ZNSt6vectorISt4pairIN7HepGeom7Point3DIdEES3_ESaIS4_EE19_M_emplace_back_auxIJS4_EEEvDpOT_	16 GB	0 B	1,934,270	libG4geometry.so
G4BoundingEnvelope::CreateListOfEdges	8 GB	0 B	1,934,270	libG4geometry.so
G4MaterialPropertiesTable::GetProperty	4 GB	0 B	140,021,992	libG4materials.so
G4BoundingEnvelope::CalculateExtent	2 GB	0 B	4,244,880	libG4geometry.so
G4ParticleHPVector::Check	2 GB	215 MB	190,145	libG4processes.so
[Others]	16 GB	1 GB	84,294,069	

Collection and Platform Info

This section provides information about this collection, including result set size and collection platform data.

Application Command Line:	
Operating System:	4.4.103-6.38_4.0.151-cray_ari_c NAME="SLES" VERSION="12-SP3" VERSION_ID="12.3" PRETTY_NAME="SUSE Linux Enterprise Server 12 SP3" ID="sles" ANSI_COLOR="0;32" CPE_NAME="cpe:/o:suse:sles:12:sp3"
Computer Name:	nid00747
Result Size:	63 GB
Collection start time:	03:02:02 28/11/2018 UTC
Collection stop time:	04:33:35 28/11/2018 UTC
Collector Type:	User-mode sampling and tracing

CPU

Name:	Intel(R) Xeon(R) E5/E7 v3 Processor code named Haswell
Frequency:	2.3 GHz
Logical CPU Count:	64

…. reducing memory usage

Find memory leaks using VTune

<no current project> - Intel VTune Amplifier

Welcome | res_dir...

Memory Consumption Memory Consumption viewpoint (change)

INTEL VTUNE AMPLIFIER 2018

Analysis Target | Analysis Type | Collection Log | Summary | Bottom-up | BaccBST.cc

Source | Assembly | Assembly grouping: Address

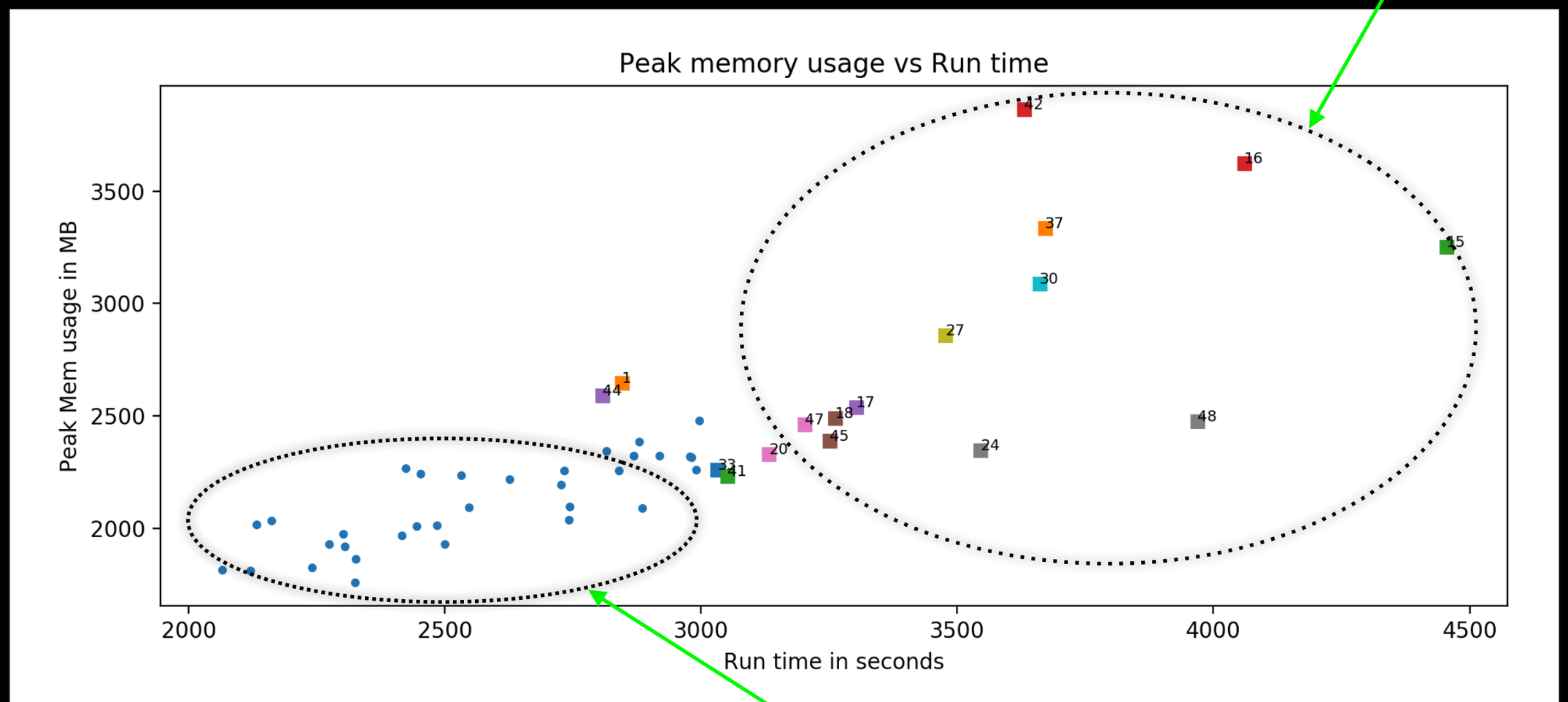
So..	Source	Alloc...	De..	Allocatio...	Source File
34	`{`				
35	`  //  Create the first level with a node in the middle of the decay time space`				
36	`  topNode->timeOfEvent = (windowEnd + windowStart) * 1.e9 * ns / 2.; //in nanoseconds`				
37					
38	`  //  Now step through each additional initialization level, creating the`				
39	`  //  other emptry nodes, always chopping the remaining space into halves.`				
40	`  //  Note that the topmost level is level 1, so we start here with level 2.`				
41	`  Isotope* tmpIso = 0;`				
42					
43	`  for (G4int i = 2; i <= numInitLevels; i++)`				
44	`  {`				
45	`  G4double timeStep = (windowEnd + windowStart) / pow(2, i) * 1.e9 * ns; //nanoseconds`				
46	`  G4double runningTime = windowStart * 1.e9 * ns + timeStep; //ns`				
47	`  G4int numPointsInLevel = (G4int)pow(2, i - 1);`				
48	`  //if(numPointsInLevel<100)std::cout << "BST nodePoint " << numPointsInLevel`				
49	`  //  << " times created: " << std::endl;`				
50	`  for (G4int j = 0; j < numPointsInLevel; j++)`				
51	`  {`				
52	`  tmpIso = new Isotope("Empty", 0, 0, 0);`	151 MB		1,656,230	BaccBST.cc
53	`  Insert(tmpIso, runningTime, G4ThreeVector(0, 0, 0), -1, -1, topNode);`				
54	`  //if(numPointsInLevel<100) std::cout << j << ":" << runningTime/ns << " ";`				
55	`  runningTime += 2 * timeStep;`				
56	`  }`				
57	`  //if(numPointsInLevel<100) std::cout << "\n" << std::endl;`				
58	`  }`				
59	`  G4cout << "Finished building empty nodes in BST" << G4endl;`				
60					
61	`  hasNodes = true;`				
62	`}`				
63					
64	`////////////////////////////////////////////////////////////////////////////////`				
65	`void BaccBST::Insert(Isotope* iso, G4double theTime, G4ThreeVector Pos, G4int sourceByVolumeID, G4int sourcesID)`				
66	`{`				
67	`  nodeCount = 0;`				
68	`  // Time sent and received in nanoseconds`				
69	`  Insert(iso, theTime, Pos, sourceByVolumeID, sourcesID, topNode);`				
70					
71	`  if (nodeCount > 1000 && warningsWritten < 10)`				
72	`  {`				
Sel..		151 MB		1,656,230	

…. reducing memory usage

- ✓ Code modified, features added
- ✓ Memory leaks plugged

Some jobs still take a lot of memory

Peak memory usage for a set of 50 jobs with different seeds

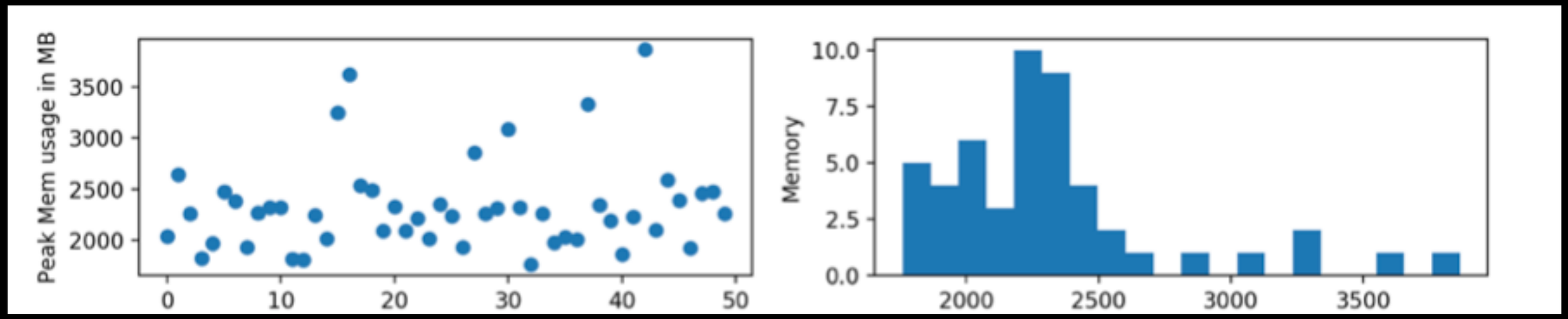


Average usage is lower

…. reducing memory usage

	September 2018	September 2019
Average memory usage	~ 4 GB	~ 2.3 GB
Jobs per node	32	50

Memory usage summary for a set of 50 jobs with different seeds
on a single node with current version of code



Run number

Memory histogram

Summary

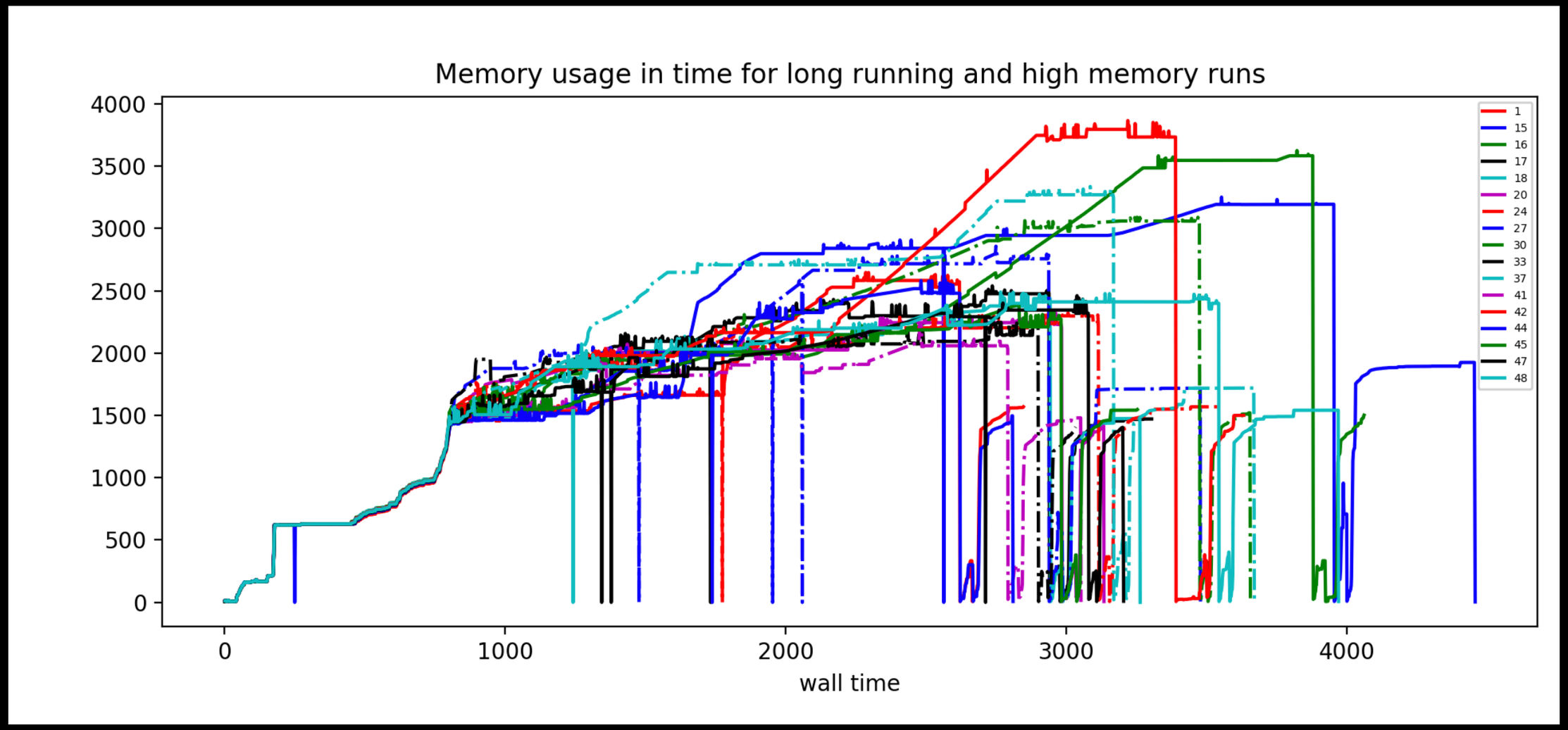
- LZ is successfully running workflow on HPC
- Increased efficiency by reducing memory footprint using tools like VTune and Process Monitor
- GUI for job submission

Future Direction

- Data collection in 2020
- Checkpointing of jobs
- Multi-threading for GEANT4

Thank you

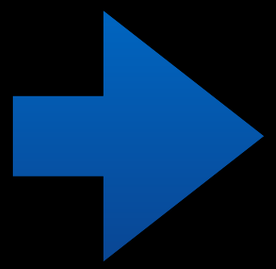
Back up slides



Some jobs still take high memory

Other challenges

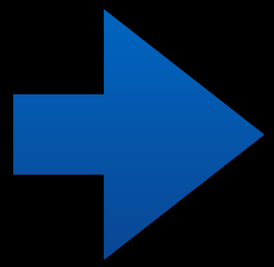
Need a custom environment for running applications



Shifter

- Custom environment for running scripts provided by NERSC
- Containers (like Docker)
- No root access necessary

cvmfs



- CERN Virtual File System
- Used to store compiled binaries for use
- Custom installation at NERSC to mount on cray machines