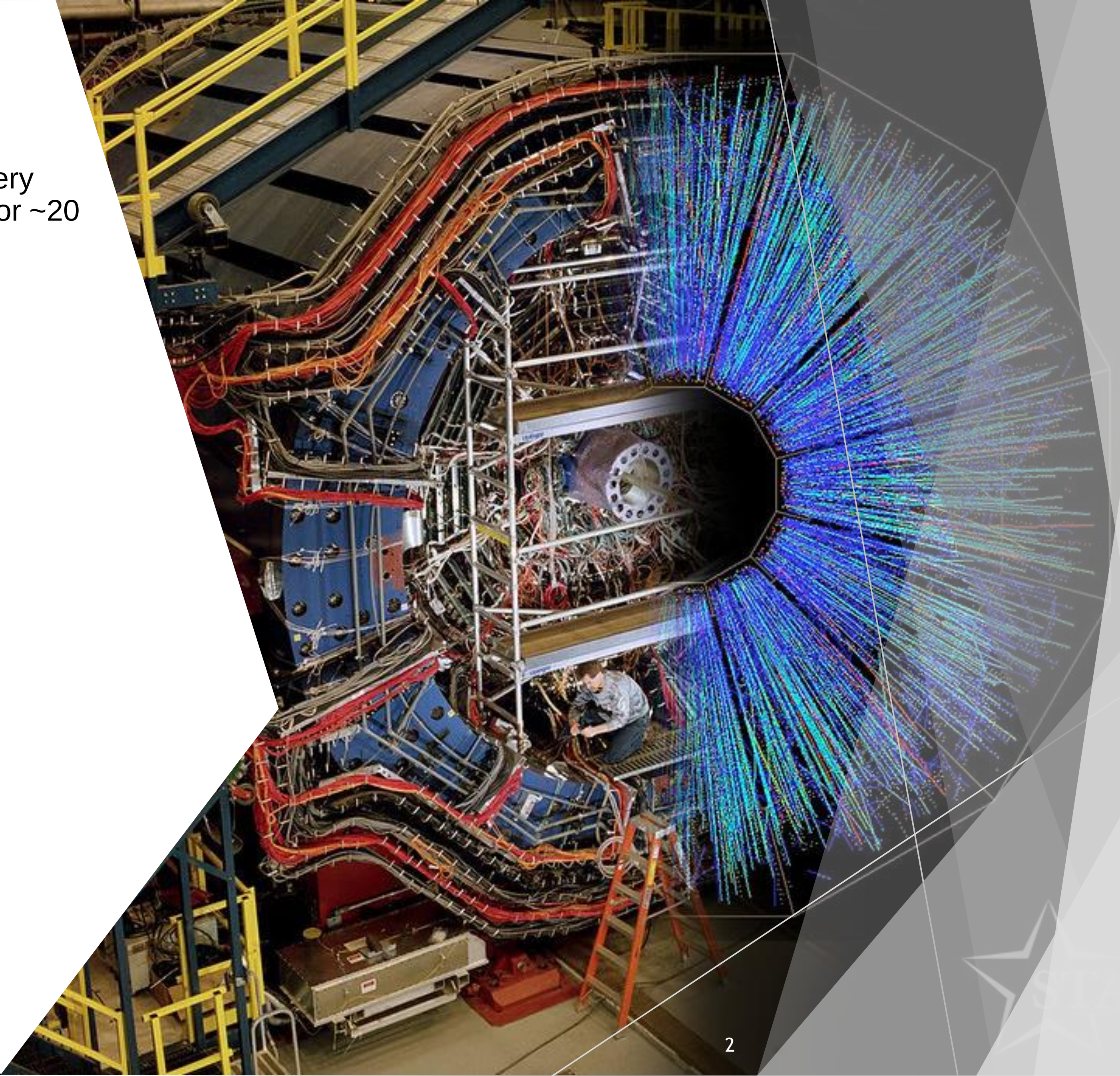
Physics Data Production on HPC: Experience to be efficiently running at scale

Michael D. Poat, Jérôme Lauret, Jefferson Porter, & Jan Balewski

NERSC Users Group SIG Annual Meeting
August 17th, 2020

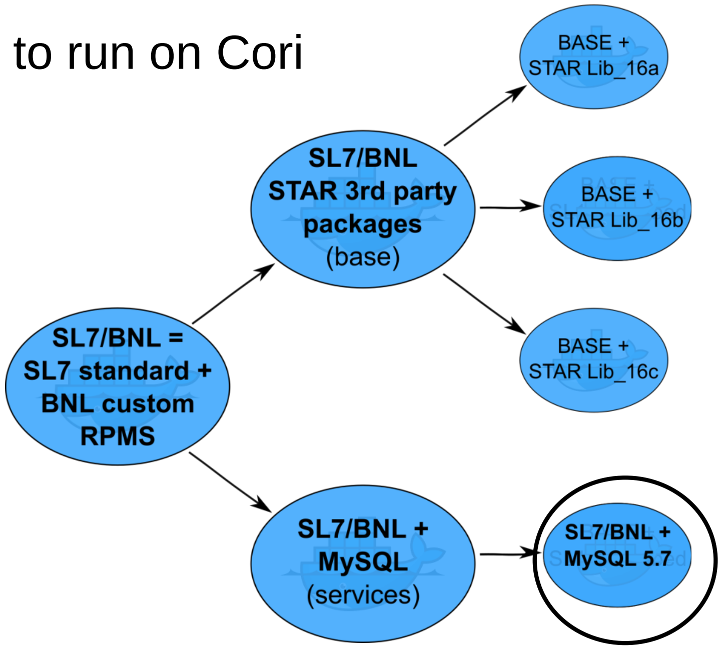
Introduction

- The STAR detector at RHIC produces 10s of PB every year and ran its data production on NERSC/PDSF for ~20 years
- PDSF’s is EOL -> migrated to NERSC/Cori
- Ongoing Efforts for STAR Data Production on Cori
  - Container Model
  - Scalability of CVMFS serving the STAR SW on Cori
  - Workflow on Cori
  - MySQL Database access
  - Efficiency



STAR Software in Containers

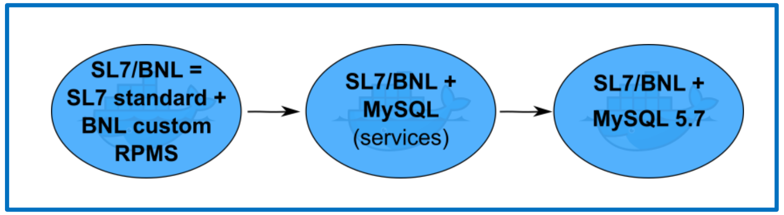
- Docker/Shifter containers are required to enable the STAR Software to run on Cori
- Best to deploy minimal containers, with Software stack provisioned from CVMFS
- Initial Container Model:
  - Base OS SL7 + RPM + STAR SW + 1 STAR Library (4 GB)
- Minimal Container Model:
  - Base OS SL7 + RPM
  - CVMFS Serves: STAR SW + STAR Libraries



SL7/BNL = SL7 standard + BNL custom RPMS
SL7/BNL STAR 3rd party packages (base)
BASE + STAR Lib_16a
BASE + STAR Lib_16b
BASE + STAR Lib_16c
SL7/BNL + MySQL (services)
SL7/BNL + MySQL 5.7

Container Maintenance Tree

- Our previous setup required 1 node to run a MySQL DB container while all other worker nodes would run STAR tasks
- The current running setup combines STAR Tasks & MySQL Database on 1 node
- Current Container: **SL7 + RPM + mysqld**



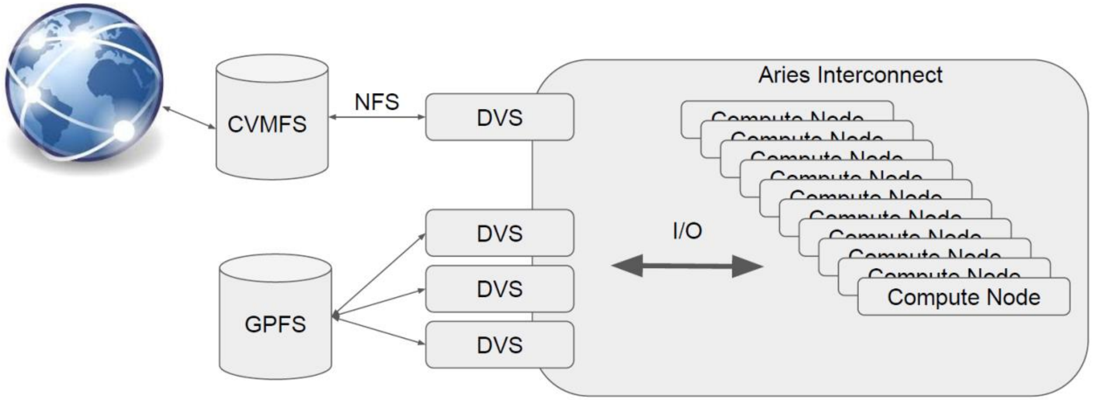
SL7/BNL = SL7 standard + BNL custom RPMS
SL7/BNL + MySQL (services)
SL7/BNL + MySQL 5.7

Current Running Setup on Cori

CVMFS on Cori

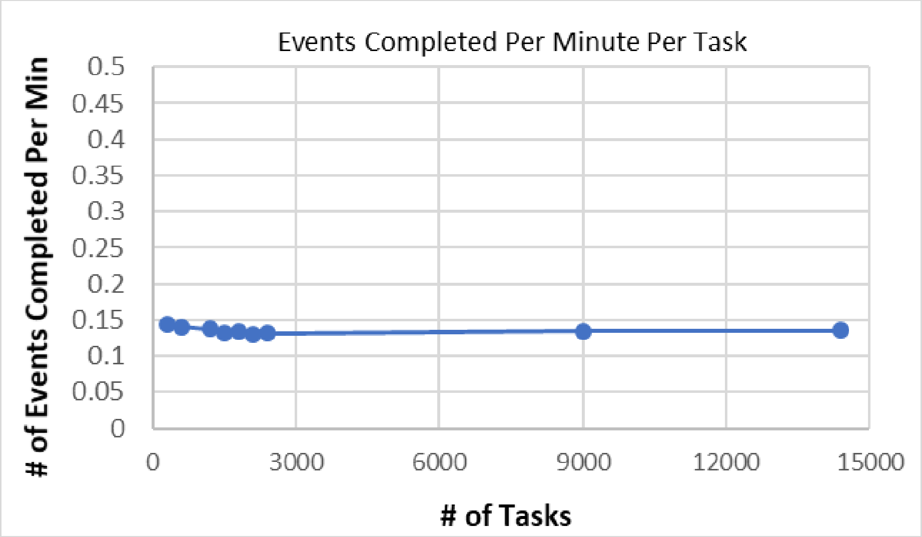
CVMFS on Cori

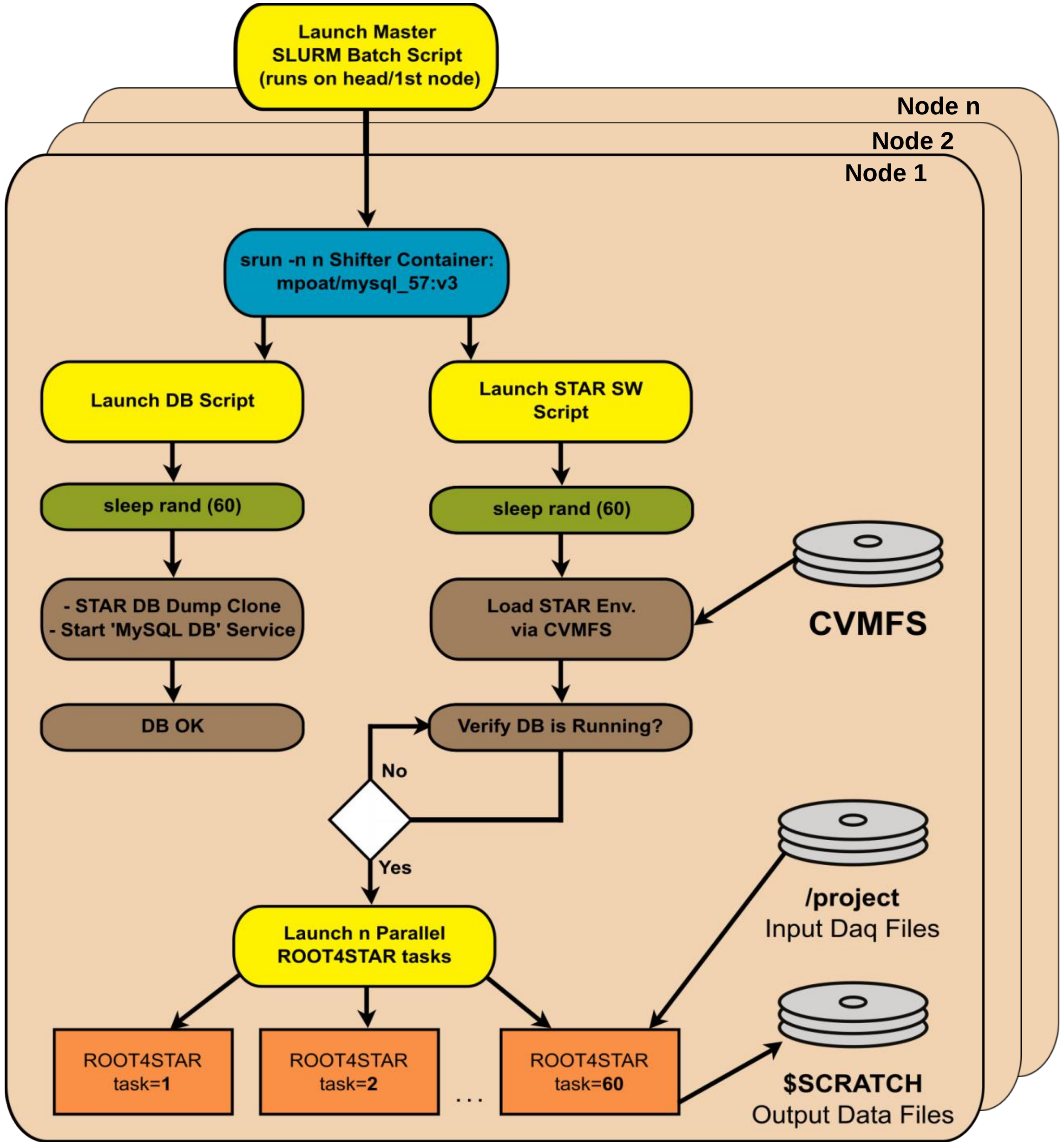
- Fuse restriction on Cori (No Kernel access on worker nodes) - cannot munt CVMFS natively
- NERSC provides Cori with Data Virtualization Service (DVS) servers
- **DVS servers forward I/O well, but do not support metadata lookups (requires lookup to real CVMFS backend -> latency)**



Throughput Maximization for CVMFS

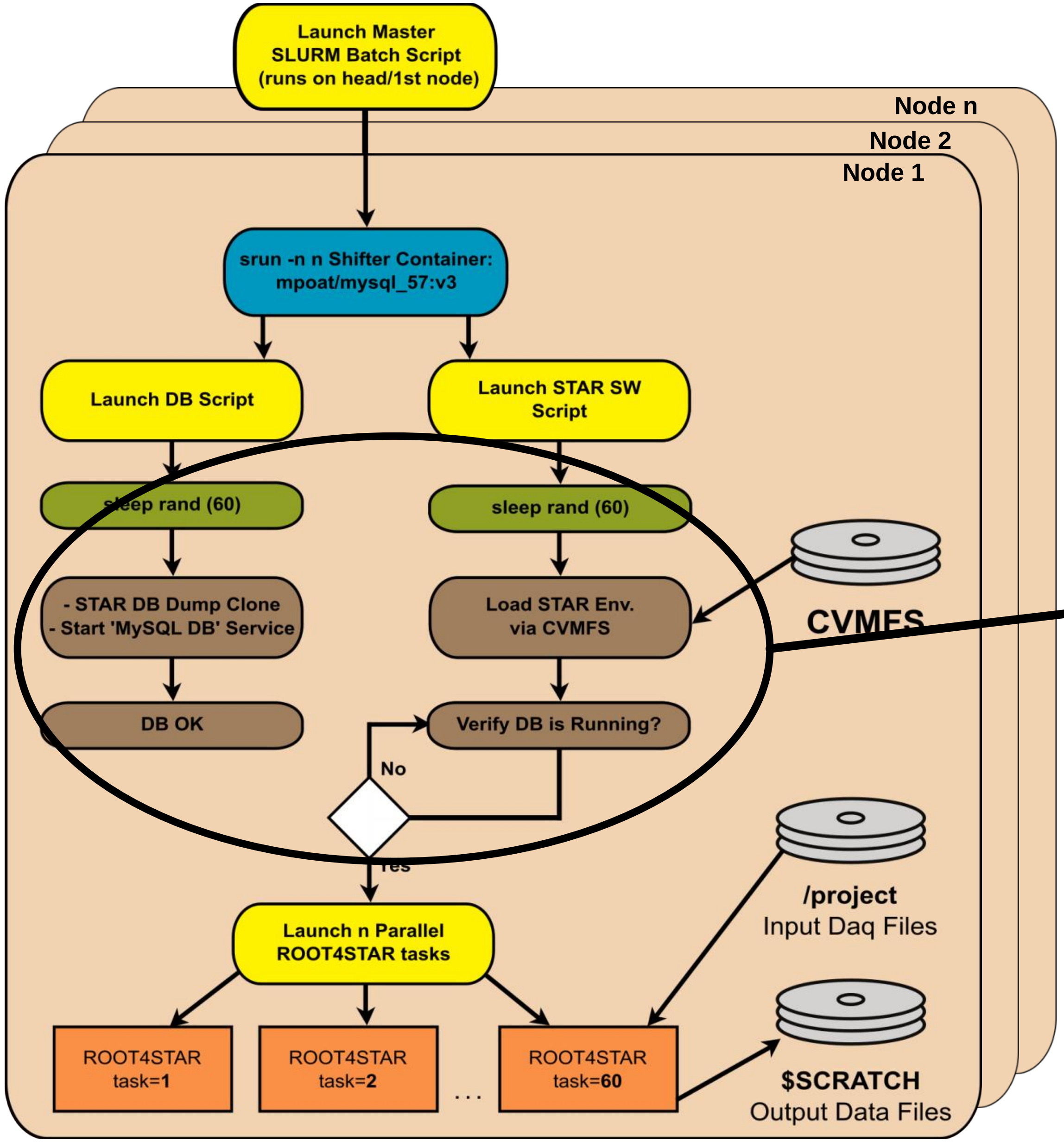
- Looked at average of events produced min/“task”
- Drops by ~10-12% at first but we still gain in “events min/node”
- Curve remains flat afterward up to our max @15,000 tasks on 240 nodes
- **In order to achieve this we needed to modify our workflow with time delays…**





STAR Workflow on Cori

- First we launch steering script to the batch system
- Starts the STAR+mysqld container
- Runs ‘Load DB’ & STAR SW scripts in parallel
- Both scripts have random sleep delays (one for copying the DB and 1 for loading SW via CVMFS)
- Once STAR SW is loaded the script will wait until the DB has started (biggest time killer!)
- Node(s) will launch ‘n’ Parallel ROOT4STAR tasks



Launch Master SLURM Batch Script (runs on head/1st node)

Node n

Node 2

Node 1

srun -n n Shifter Container: mpoat/mysql_57:v3

Launch DB Script

Launch STAR SW Script

sleep rand (60)

sleep rand (60)

- STAR DB Dump Clone
- Start 'MySQL DB' Service

Load STAR Env. via CVMFS

CVMFS

DB OK

Verify DB is Running?

No

Yes

Launch n Parallel ROOT4STAR tasks

ROOT4STAR task=**1**

ROOT4STAR task=**2**

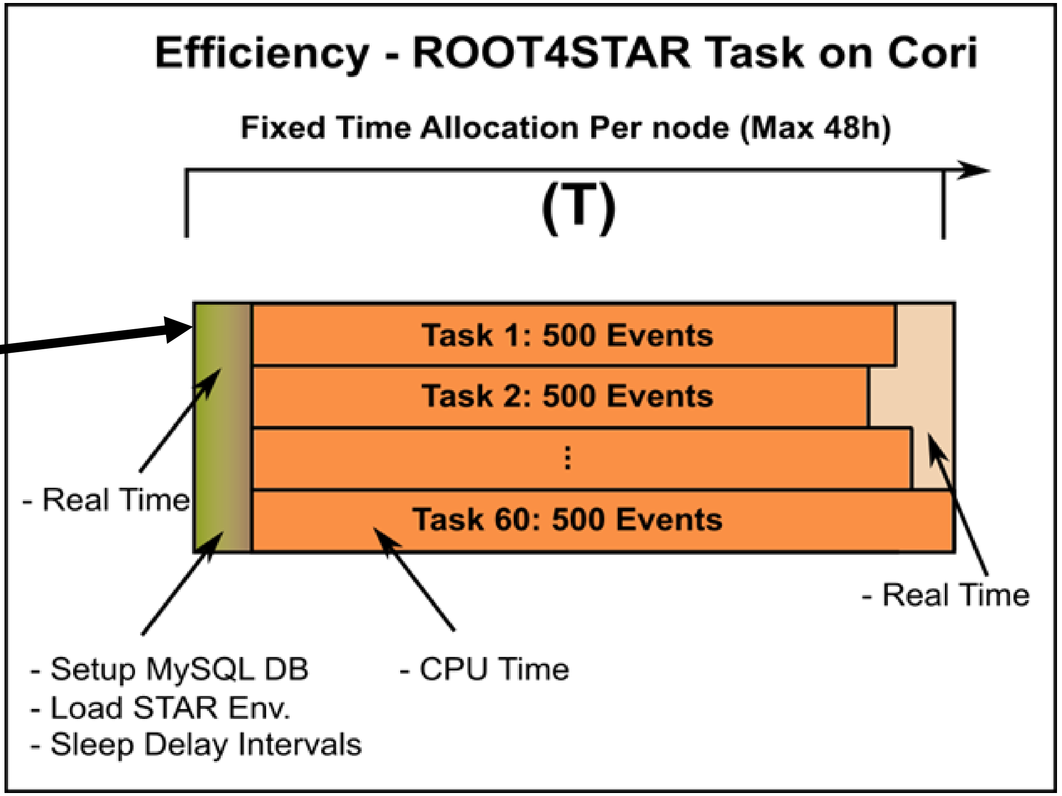
. . .

ROOT4STAR task=**60**

/project
Input Daq Files

$SCRATCH
Output Data Files

Job start efficiency loss



Efficiency - ROOT4STAR Task on Cori

Fixed Time Allocation Per node (Max 48h)

(T)

Task 1: 500 Events

Task 2: 500 Events

⋮

Task 60: 500 Events

- Real Time

- Real Time

- Setup MySQL DB
- Load STAR Env.
- Sleep Delay Intervals

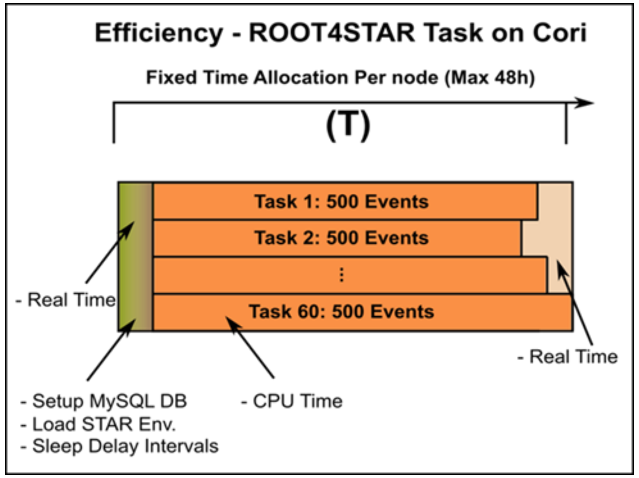
- CPU Time

Efficiency on Cori

Goal: Maximize (event per sec. / per $)

- Dedicating 1 head node as DB only to serve 10 worker nodes (1-to-11) **VS.** (1-to 1) model (each worker node self-serves DB)
  - 1-to-1 model: Total Eff. 99.30%
  - 1-to-11 model: Total Eff. 89.44%
  - **Better to self-serve DB**
- Job Start Efficiency: we lose ~.05%
- Event Efficiency: ~98-99%
  big job = highest value
- Total Efficiency on 1-to-1 KNL/Haswell, and BNL BCF: ~98-99%
- Total vCore Utilization:
  - Haswell: 87% @ 60 task + 1 DB
  - KNL: 36.9% @ 100 task + 1 DB
  - Cannot maximize CPU util. due to memory limit -> **Best to focus on packing best # of tasks per/node & Total Efficiency**

- **Job Start Efficiency**: Real time to copy/start DB, load env., sleep delays **(E1)**
- **Event Efficiency**: CPU/Real time ratio for STAR event data reconstruction **(E2)**
- **Total Efficiency**: SLURM job Start ->Last Task Finished (NodesUsed/NodesUnused) * E1 * E2



Efficiency - ROOT4STAR Task on Cori

Fixed Time Allocation Per node (Max 48h)

Job	**(T) DB dump, Load Env., Rand (1-60s) delays**	**Job Start Efficiency** (Total Job Time - (T))/Total Job Time **(E1)**	**Event Efficiency** All Events **(E2)**	**Total Efficiency** (NodesUsed/Nodes Unused) * E1 * E2
KNL 1 Node (Long Test - 60 task)	819 sec.	99.50%	99.79%	**99.30%**
KNL 11 Nodes 1 Node ded. DB server (60 task)	864 sec.	99.48%	99.90%	**89.44%**
Haswell 1 Node (Long Test - 60 task)	378 sec.	99.76%	99.04%	**98.80%**
BNL RCF Job - 100 tasks	1 sec.	99.99%	99.81%	**98.82%**

Conclusion

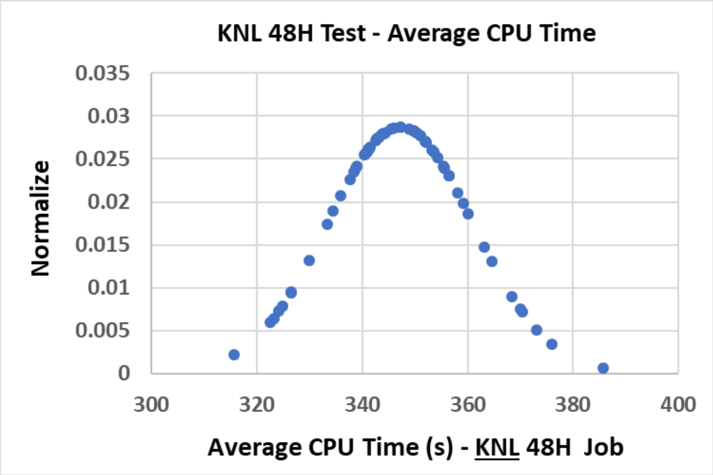
- **Docker/CVMFS:**
  - Containers are kept to minimum -> SL7 + RPM + mysqld, Software provisioned from CVMFS via DVS servers on Cori
- **Database:**
  - DB can be copied to NERSC on demand and remerged with authentication tables
  - On Cori: Worker node running ‘mysqld’ DB instance + R4S tasks to self-serve & serve DB connections to some worker nodes -> most efficient model
- **Workflow:**
  - Launch DB & environment scripts in parallel
  - Time delays required (latency) for CVMFS via DVS
- **Efficiency:**
  - “Job Start Efficiency” and Idle CPU at the end of job have minimal impacts on “Total CPU/Real time Efficiency” if we run for maximize node allocation (48h)
  - Head node model introduces biggest efficiency % loss
  - Haswell provides best CPU power / $ for us

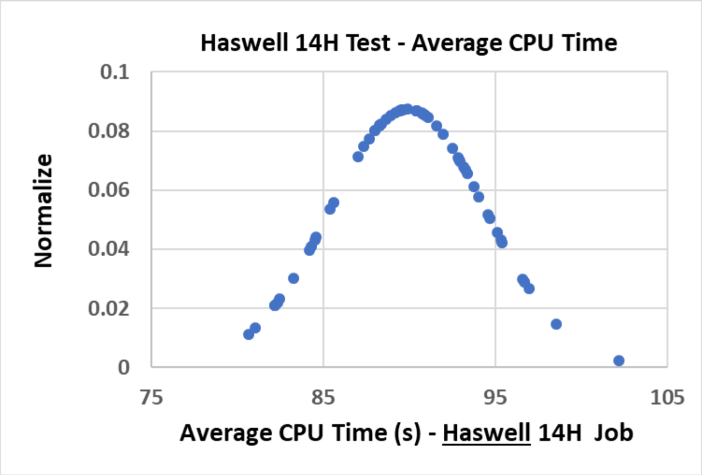
Our next steps

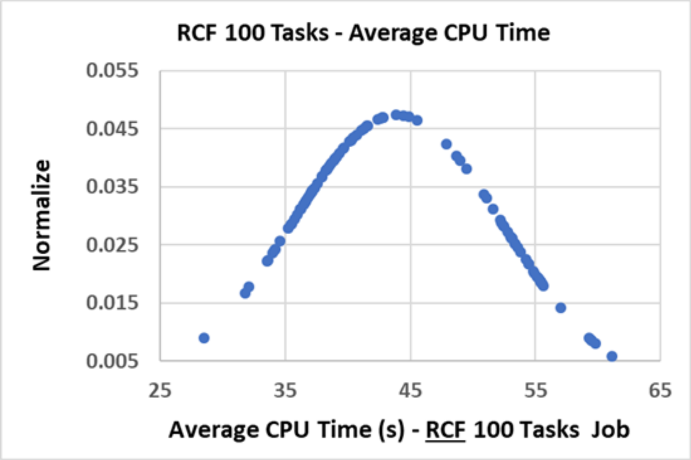
- Ensure graceful termination of the tasks (use of “signal handling”)
- Potential use of Burst Buffer to pre-stage DB content
- “Event Service” is coming soon

Thanks!

Throughput Estimator







- Due to the efficiency loss at the start & end of a job, it is best to run for the maximum amount of time (48h)
- By obtaining the average time events are processed per task, we can estimate how long a job will take
  - Multiple tests run on a single **KNL** node**,** a single **Haswell** node**,** & **BNL RCF (2.8GHz Intel)**
- The distribution and scaling is very predictable between the systems on any dataset
  - With the estimator, we only need to run a small batch of jobs on our BNL RCF farm to get estimate of total time on Cori KNL/Haswell
- **Provides starting point for an “Event Service” to launch new tasks when one finishes**