



HTCondor at BNL SDCC

Kevin Casella
kac@bnl.gov

Tom Smith
tsmith@bnl.gov

Christopher Hollowell
hollowec@bnl.gov

13 July 2023 – Throughput Computing 2023

    [@BrookhavenLab](https://twitter.com/BrookhavenLab)

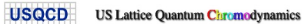
SDCC: The Scientific Data and Computing Center

- Located at Brookhaven National Laboratory (BNL) on Long Island, New York
- SDCC initially formed at BNL in the mid-1990s as the RHIC Computing Facility



Shared multi-program facility serving ~2,000 users
from more than 20 projects

How much data do you have?



Scientific Data and Computing Center Overview

- Tier-0 computing center for the RHIC experiments
- US Tier-1 Computing facility for the ATLAS experiment at the LHC, also one of the ATLAS shared analysis (Tier-3) facilities in the US
- Computing facility for NSLS-II
- US Data center for Belle II experiment
- Providing computing and storage for proto-DUNE/DUNE along w/ FNAL serving data to all DUNE OSG sites
- Also providing computing resources for various smaller / R&D experiments in NP and HEP
- Serving more than **2,000** users from **> 20 projects**
- Developing and providing administrative/collaborative tools:
 - Invenio, Jupyter, BNL Box, Discourse, Gitea, Mattermost, etc.
- sPHENIX - commissioning run has started
- BNL was selected as the site for the upcoming major new facility Electron-Ion Collider (EIC/eRHIC)



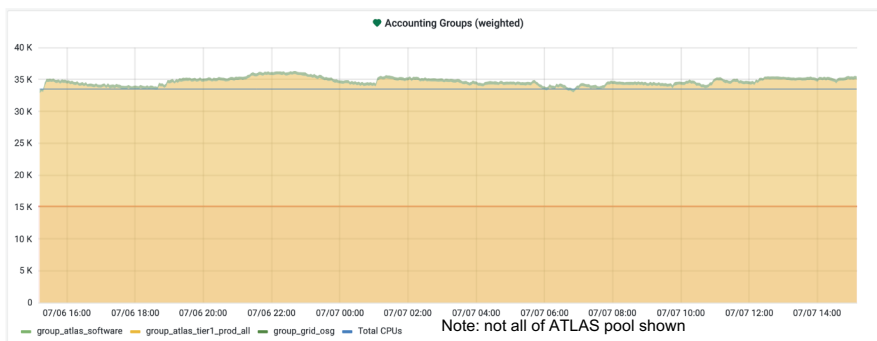
High Throughput Computing @ SDCC

- Providing our users with ~1,700 HTC nodes:
 - ~121,000 logical cores
 - Managed by HTCondor
- All nodes running Scientific Linux (SL) 7
 - Preparations for an OS upgrade to Alma Linux 9 in progress
- HTCondor 9.0.X -> 10.0.X upgrade in progress



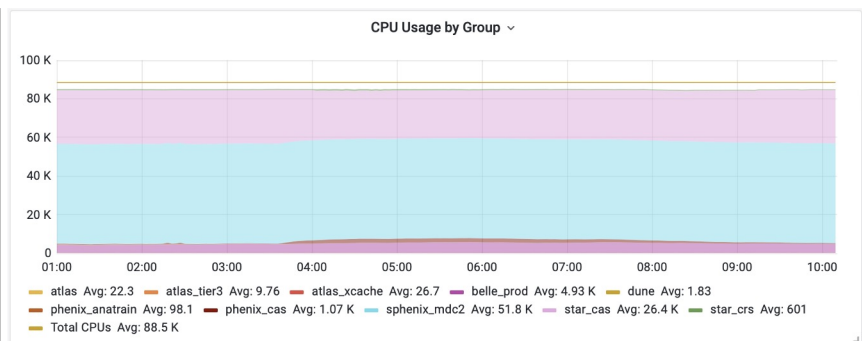
ATLAS Pool

- ~500 HTC nodes:
 - ~35,000 logical cores
- ATLAS tier 1 worker nodes upgraded to Condor 10.0.3



Shared Pool

- ~1,200 HTC nodes:
 - ~86,000 logical cores
- Large portion of this pool is new Supermicro nodes



Computing Expansion (2023)

- Largest computing expansion to date
- Purchased 648 Supermicro SYS-610C-TR nodes for ATLAS and the RHIC experiments (~62k logical cores total)
 - Housed in 20 racks
 - System specs:
 - Dual Intel Ice Lake Xeon Gold 6336Y 24-core processors
 - 12x32 GB 3200 MHz ECC DDR4 RAM (384 GB total)
 - 4x2 TB SSD drives
 - 1U form factor
 - 10 Gbps NIC
- Will be receiving some Supermicro ARM test nodes Aug/Sep
 - With Ampere Altra CPUs
- All nodes running Scientific Linux (SL) 7
 - Condor 10.0.X installed



Supermicro SYS-6019U-TR4 Servers

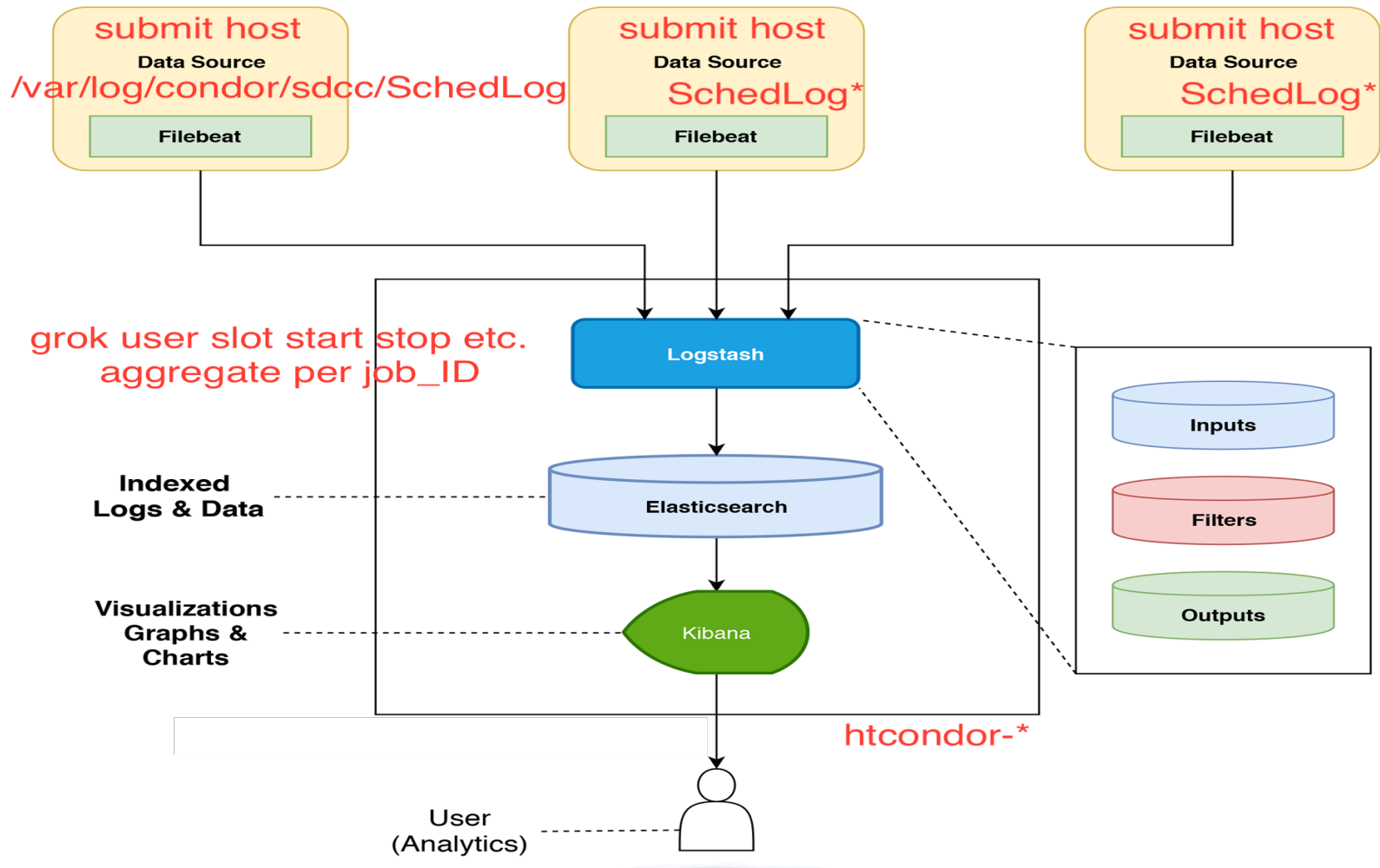
Condor/ Condor CE Upgrade

- Upgrade from Condor 9.0.x to 10.0.X and Condor CE from 5.1.5-1 to 6.X
- To minimize downtime, we upgrade in systems in batches of several racks at a time
- Worker/execute nodes upgraded first, then CEs and submit nodes, and collector/negotiator will be upgraded last
- Cross-version compatibility was tested to ensure this would not impact production

collector / negotiator	schedd (interactive submit)	schedd / Condor CE (grid submit)	startd (execute nodes)
9.0.x	9.0.x	9.0.x (condor) 5.1.5-1 (CE)	Both 9.0.x / 10.0.1
9.0.x	9.0.x	10.0.1 (condor) 5.1.6-1 (CE)	Both 9.0.x / 10.0.1
9.0.x	10.0.1	10.0.1 (condor) 5.1.6-1 (CE)	Both 9.0.x / 10.0.1
10.0.1	10.0.1	10.0.1 (condor) 5.1.6-1 (CE)	Both 9.0.x / 10.0.1

- **Done:** ATLAS Tier1 worker nodes, new Supermicro shared pool worker nodes, next generation CE test system
- **In Progress:** ATLAS/grid CEs upgrade, shared pool worker nodes
- **To do:** Shared pool CEs, all pool collector/negotiators, all interactive submit nodes

Monitoring Redesign for Scalability



Monitoring Redesign for scalability

- Filebeat service runs on CEs and Submit nodes (~100 hosts)
 - A Logstash instance runs on one server
 - Input source is the log file `/var/log/condor/sdcc/SchedLog`
 - Output sent to 3 servers running Elasticsearch service
- Filebeat service previously ran on workernodes (2,000+ hosts)
 - This required multi-Logstash and the Lumberjack plugin
 - Lumberjack transported events between Logstash instances

Grok Filter Plugin

The screenshot shows the Elastic Dev Tools interface in a web browser. The address bar shows ".bnl.gov". The page has a dark header with the Elastic logo and a search bar. Below the header, there's a sidebar with "Dev Tools" selected. The main content area is titled "Sample Data" and shows a log entry: "09/02/21 13:45:57.702 (pid:1890) Started shadow for job 5471877.6 on slot1@[redacted].bnl.gov <[redacted]:9618?addr=[redacted]-9618". Below this is the "Grok Pattern" section with the pattern: "%{DATE:date} %{TIME:time} \\(%{DATA:pid}\\) Started shadow for job %{BASE10NUM:jobID} on %{EMAILADDRESS:slot} %{NOTSPACE:IP_blob} for %{NOTSPACE:u". A "Custom Patterns" section is also visible. A blue "Simulate" button is located below the pattern section. The "Structured Data" section shows the resulting JSON output: {"date": "09/02/21", "user_and_group": "group_star.cas.nish", "jobID": "5471877.6", "shadow_PID": "(shadow pid = 11107)", "IP_blob": "<[redacted]:9618?addr=[redacted]-9618&noUDP&sock=18020_33b6_3>#1628806802#17433#...", "pid": "pid:1890", "time": "13:45:57.702", "slot": "slot1@[redacted].bnl.gov"}

elastic Search Elastic

Dev Tools

Sample Data

```
1 09/02/21 13:45:57.702 (pid:1890) Started shadow for job 5471877.6 on slot1@[redacted].bnl.gov <[redacted]:9618?addr=[redacted]-9618
```

Grok Pattern

```
1 %{DATE:date} %{TIME:time} \\(%{DATA:pid}\\) Started shadow for job %{BASE10NUM:jobID} on %{EMAILADDRESS:slot} %{NOTSPACE:IP_blob} for %{NOTSPACE:u
```

> Custom Patterns

Simulate

Structured Data

```
1 {
2   "date": "09/02/21",
3   "user_and_group": "group_star.cas.nish",
4   "jobID": "5471877.6",
5   "shadow_PID": "(shadow pid = 11107)",
6   "IP_blob": "<[redacted]:9618?addr=[redacted]-9618&noUDP&sock=18020_33b6_3>#1628806802#17433#...",
7   "pid": "pid:1890",
8   "time": "13:45:57.702",
9   "slot": "slot1@[redacted].bnl.gov"
10 }
```

START of a job

```
aggregate {  
  task_id => "%{jobID}"  
  code => "  
    map['user_and_group'] ||= event.get('user_and_group')  
    map['slot'] ||= event.get('slot')  
    map['execution_host'] ||= event.get('execution_host')  
    map['user'] ||= event.get('user')  
  "  
  map_action => "create"  
}
```

Create a map

Map event data

END of a job

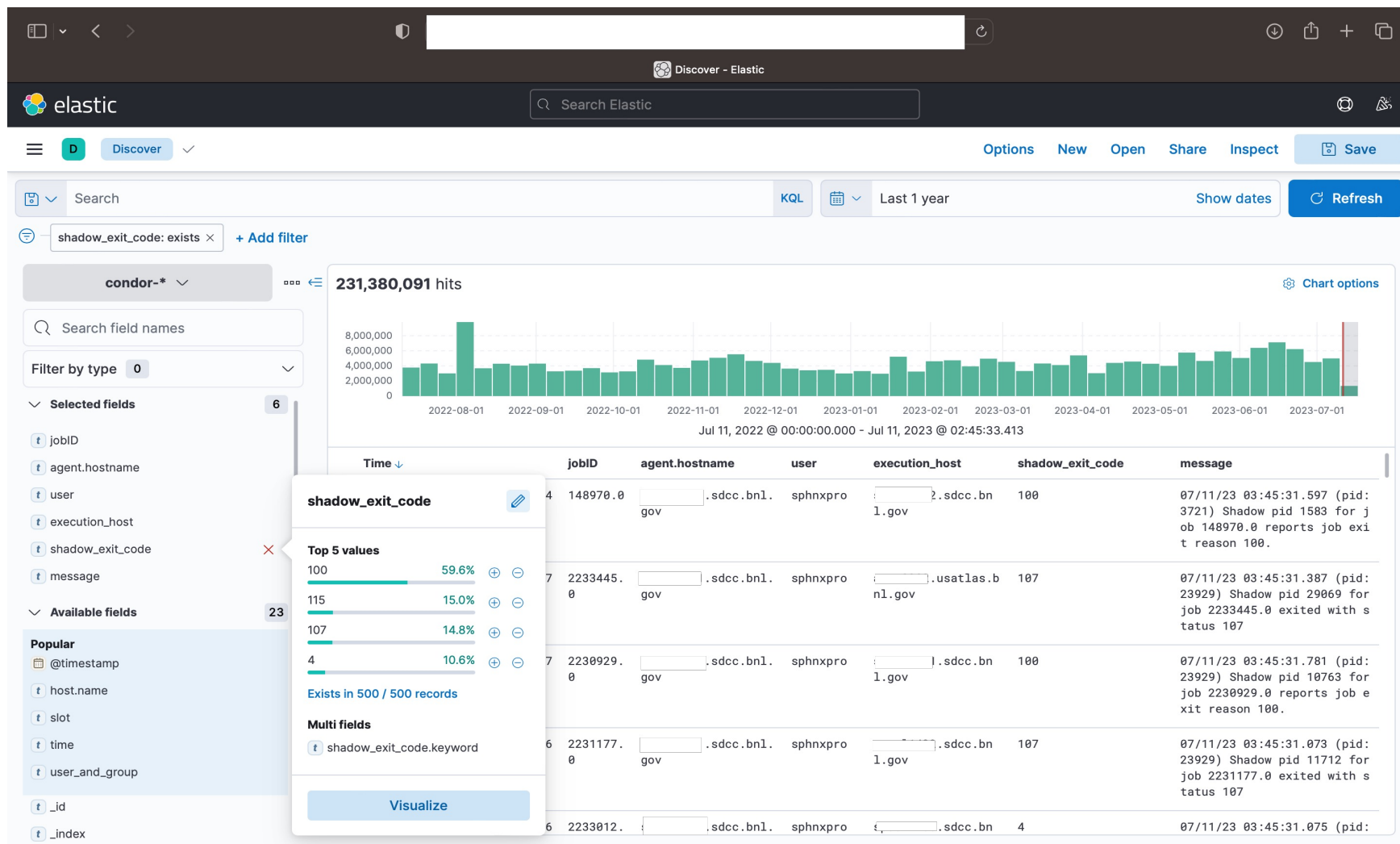
```
aggregate {  
  task_id => "%{jobID}"  
  code => "  
    event.set('user_and_group', map['user_and_group'])  
    event.set('slot', map['slot'])  
    event.set('execution_host', map['execution_host'])  
    event.set('user', map['user'])  
  "  
  map_action => "update"  
  end_of_task => true  
  timeout => 604800  
}
```

Set event fields

From the map

Delete the map

Kibana Discover



Conclusions

- SDCC provides a batch computing capacity of over 120k cores to 2,000+ users in more than 20 projects
 - We've been using HTCondor for nearly 20 years
- HTCondor 10.0.x upgrade in progress
 - Upgrading a rolling manner to minimize downtime
 - Completed on all US ATLAS T1 compute nodes
- We've implemented custom ELK monitoring of Condor jobs at our facility
 - Recently re-designed to improve scalability
 - Changed from collecting data on all workernodes to submit nodes only

Questions & Comments

Kevin Casella
kac@bnl.gov

Tom Smith
tsmith@bnl.gov

Christopher Hollowell
hollowec@bnl.gov

13 July 2023 – Throughput Computing 2023



@BrookhavenLab