
Benchmarking the Processing of Aircraft Tracks with Triples Mode and Self-Scheduling

Andrew Weinert, Marc Brittain, Ngaire Underhill, Christine Serres

IEEE High Performance Extreme Computing (HPEC) Conference

September 2021

This material is based upon work supported by the Federal Aviation Administration under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Aeronautics and Space Administration.



DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

© 2021 Massachusetts Institute of Technology.

Delivered to the U.S. Government with Unlimited Rights, as defined in DFARS Part 252.227-7013 or 7014 (Feb 2014). Notwithstanding any copyright notice, U.S. Government rights in this work are defined by DFARS 252.227-7013 or DFARS 252.227-7014 as detailed above. Use of this work other than as specifically authorized by the U.S. Government may violate any copyrights that exist in this work.



FAA Disclaimer

This document is derived from work done for the FAA (and possibly others), it is not the direct product of work done for the FAA. The information provided herein may include content supplied by third parties. Although the data and information contained herein has been produced or processed from sources believed to be reliable, the Federal Aviation Administration makes no warranty, expressed or implied, regarding the accuracy, adequacy, completeness, legality, reliability or usefulness of any information, conclusions or recommendations provided herein. Distribution of the information contained herein does not constitute an endorsement or warranty of the data or information provided herein by the Federal Aviation Administration or the U.S. Department of Transportation. Neither the Federal Aviation Administration nor the U.S. Department of Transportation shall be held liable for any improper or incorrect use of the information contained herein and assumes no responsibility for anyone's use of the information. The Federal Aviation Administration and U.S. Department of Transportation shall not be liable for any claim for any loss, harm, or other damages arising from access to or use of data or information, including without limitation any direct, indirect, incidental, exemplary, special or consequential damages, even if advised of the possibility of such damages. The Federal Aviation Administration shall not be liable to anyone for any decision made or action taken, or not taken, in reliance on the information contained herein.



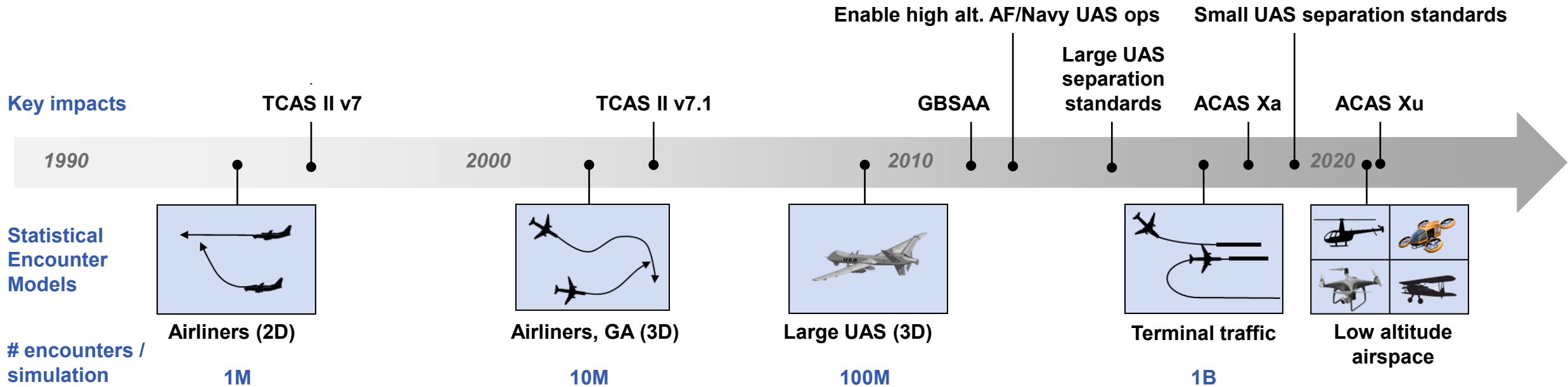
Encounter Models Enable Safety Evaluation of Collision Avoidance Systems

- **Aircraft must operate as to not create a MAC hazard due to the loss of life and property**
- **Collision avoidance systems are mandated to minimize the MAC risk between aircraft**
 - **Systems must meet sets of safety and suitability performance requirements**
 - **Separation criteria can be dependent upon the types of encountering aircraft**
- **Fast-time modeling and simulation routinely used to evaluate these systems**
- **Aircraft behavior often represented by statistical encounter models**
 - **Different models available based on specific aircraft types of operations**
 - **Historically trained via classical machine learning with surveilled aircraft tracks**
- **ASTM F38 and RTCA Special Committees 147 and 228 are developing performance standards to evaluate and certify collision avoidance systems**
 - **Standards development activities consider different UAS concepts of operations and scope**
 - **Encounter models for low altitude and terminal environments are a priority**



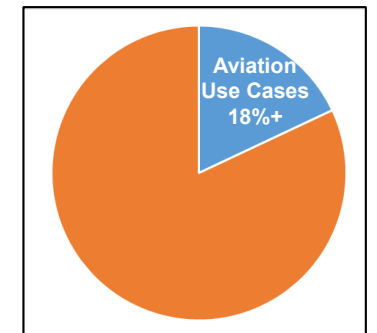
Airspace Modeling and Simulation

MIT Lincoln Laboratory Encounter Models



Key model requirements

- Accurate dynamic behaviors and flight characteristics
- Statistically-representative types of encounters (head-on, turning, climbing, etc.)
- Sufficient quantities to include low-probability events

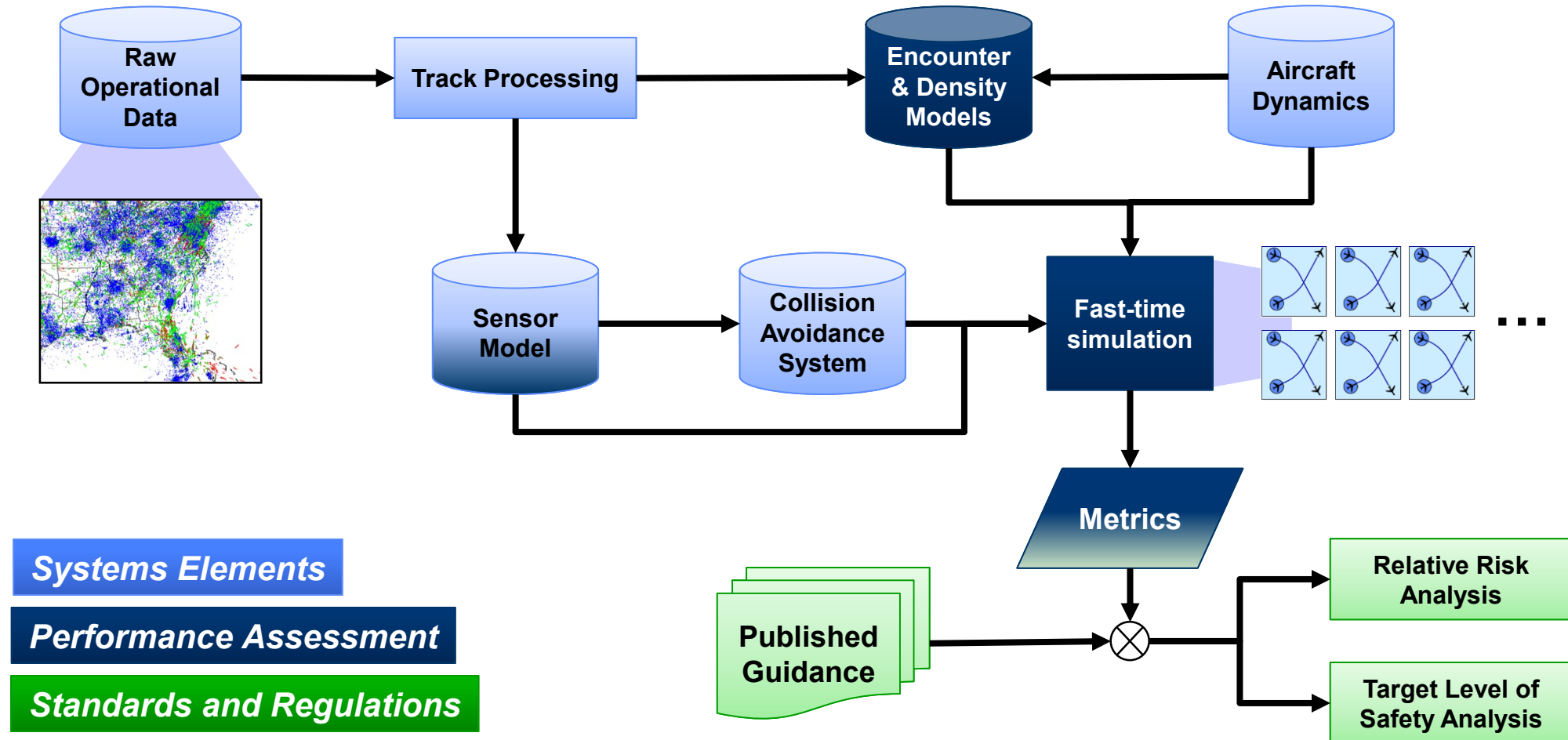


Tx-Green 2020 Usage

Growth in complexity of analyses has required growth in model sophistication and HPC resources



Example Assessment Framework

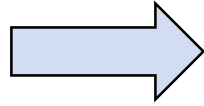




Objectives and Contributions

- **Previously* designed and prototyped a HPC workflow for model development**
 - Emphasis on efficient storage and directory hierarchy
 - Establish end-to-end processing prototype
 - Workflow consists of four steps: download, organize, archive and process
- **Simple batch distribution parallel job management resulted in poor load balancing**
 - Required 7-14 days to fully execute workflow
 - Hindered capability to conduct rapid or comprehensive analyses
 - As more data becomes available, time management issue would worsen
- **In response, developed a significantly more efficient HPC job and task allocation**
 - Benchmarked capabilities with two different aviation datasets
 - Leveraged multiple MIT LL HPC resources and unique HPC capabilities

Time to fully execute workflow reduced from weeks to days



- **Introduction**
- **Data Source**
- **High Performance Computing Resources**
- **Benchmark Results**
- **Other Improvements**
- **Conclusion**

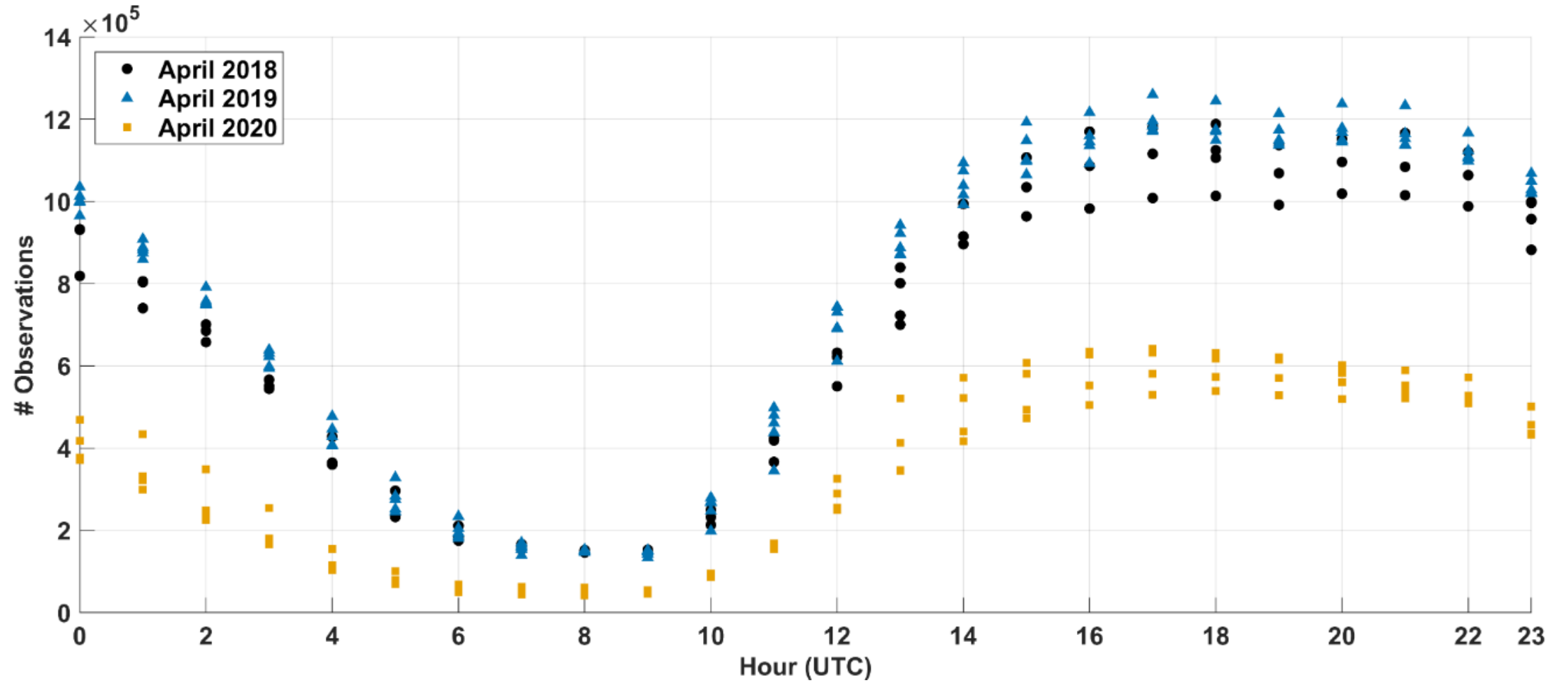


Data Source: OpenSky Network

- **Community-based receiver network which continuously collects air traffic surveillance**
 - Archives and makes accessible abstracted raw aircraft state data to researchers
 - Eight trillion+ ADS-B and Mode S messages collected from more than 1000 global sensors
 - 40 million+ daily worldwide ADS-B messages recorded from aircraft transponder broadcasts
- **MIT LL developed two datasets to train statistical models of aircraft behavior**
- **Dataset #1: 104 Mondays globally**
 - Global scope with at least 10 seconds between observations
 - No altitude filtering
 - 714 Gigabytes, across 2045 files, organized by day and hour
- **Dataset #2: 196 days near aerodromes**
 - Scope limited to within USA aerodromes, with at least 1 second between observations
 - Altitude limited based on observed MSL and estimated AGL altitudes
 - 847 Gigabytes, across 136,884 files, organized by day and location



Example Temporal Distribution





Dataset #1: 104 Mondays Globally

2425 files / 714 Gigabytes

- The OpenSky Network weekly makes easily accessible the global, abstracted, raw state data from the most recent 10-15 Mondays (UTC)
 - Each day organized into 24 files, each corresponding to one UTC hour
 - Observations at least ten second apart (either due to sampling or lack of observations)
 - No guarantee on data availability, such as if all 24 hours of a day are always available
- Data aggregated on the LLSC since 2018, but not continuously

104 Mondays For Presented Analysis

2018-02-05	2018-04-23	2018-10-29	2019-01-21	2019-04-08	2019-07-01	2019-09-16	2020-05-04	2020-07-20	2020-11-02	2021-01-11	2021-03-29	2021-06-14
2018-02-12	2018-05-14	2018-11-05	2019-01-28	2019-04-15	2019-07-08	2019-09-30	2020-05-11	2020-08-17	2020-11-09	2021-01-18	2021-04-05	2021-06-21
2018-02-19	2018-05-21	2018-11-12	2019-02-04	2019-04-22	2019-07-15	2019-10-07	2020-05-18	2020-08-24	2020-11-16	2021-01-25	2021-04-12	2021-06-28
2018-02-26	2018-09-03	2018-11-26	2019-02-11	2019-04-29	2019-07-22	2019-10-14	2020-05-25	2020-09-07	2020-11-23	2021-02-01	2021-04-19	2021-07-05
2018-03-05	2018-09-10	2018-12-03	2019-02-18	2019-05-06	2019-07-29	2020-03-16	2020-06-01	2020-09-14	2020-11-30	2021-02-08	2021-04-26	2021-07-12
2018-03-12	2018-09-17	2018-12-10	2019-02-25	2019-05-13	2019-08-05	2020-03-23	2020-06-08	2020-09-21	2020-12-07	2021-02-15	2021-05-03	2021-07-19
2018-03-19	2018-09-24	2018-12-17	2019-03-04	2019-05-20	2019-08-12	2020-03-30	2020-06-15	2020-09-28	2020-12-14	2021-02-22	2021-05-10	2021-07-26
2018-03-26	2018-10-01	2018-12-24	2019-03-11	2019-06-03	2019-08-19	2020-04-06	2020-06-22	2020-10-05	2020-12-21	2021-03-01	2021-05-17	2021-08-02
2018-04-02	2018-10-08	2018-12-31	2019-03-18	2019-06-10	2019-08-26	2020-04-13	2020-06-29	2020-10-12	2020-12-21	2021-03-08	2021-05-24	
2018-04-09	2018-10-15	2019-01-07	2019-03-25	2019-06-17	2019-09-02	2020-04-20	2020-07-06	2020-10-19	2020-12-28	2021-03-15	2021-05-31	
2018-04-16	2018-10-22	2019-01-14	2019-04-01	2019-06-24	2019-09-09	2020-04-27	2020-07-13	2020-10-26	2021-01-04	2021-03-22	2021-06-07	

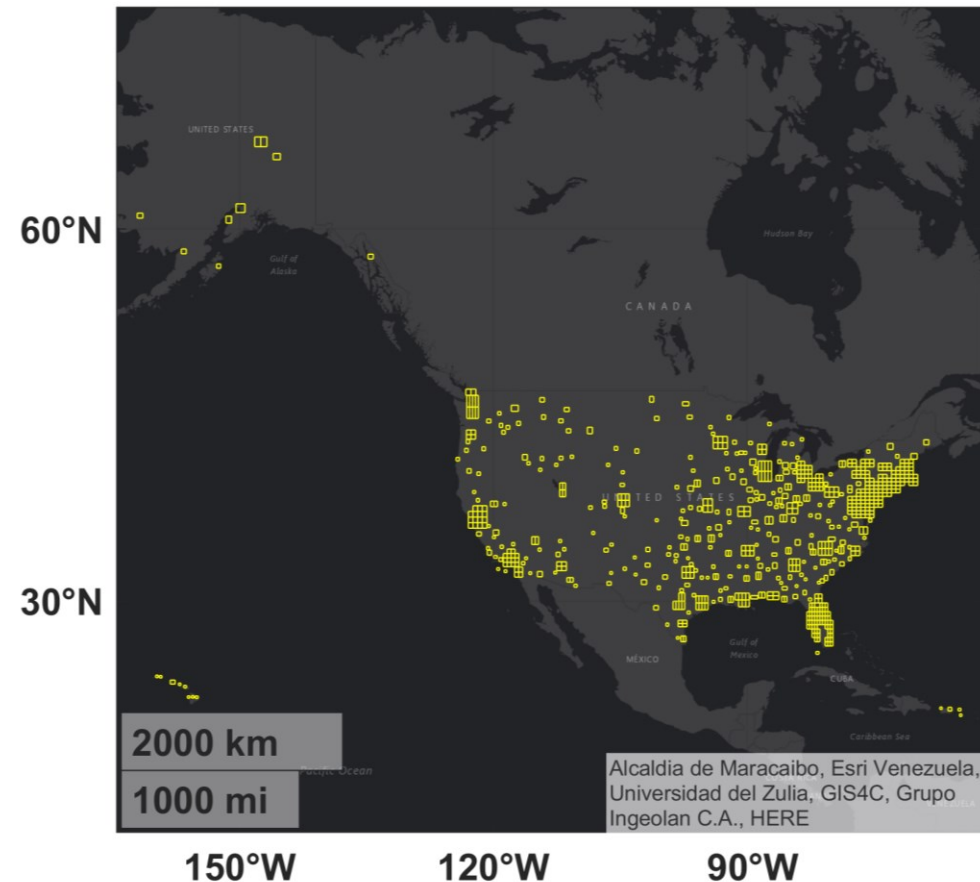


Dataset #2: 190+ Days Near USA Aerodromes

136,884 files / 847 Gigabytes

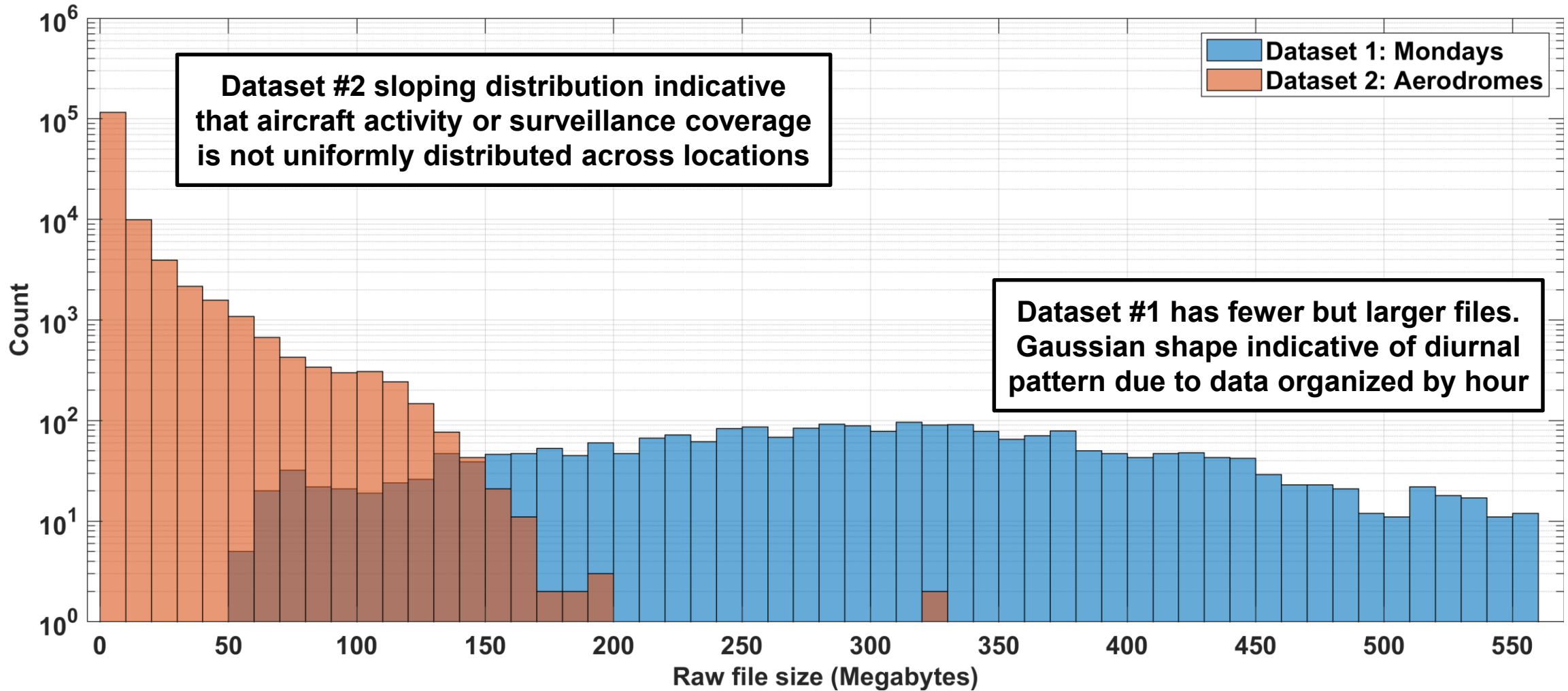
- Queried OpenSky Network Impala database based on MSL altitude, local time, latitude, and longitude
 - Organized into 695 queries, each based on a bounding box
 - Observations can be one second apart
- Targeted queries to cover airspace ≥ 8 nm away from aerodromes in FAA Class B, C, D airspaces
 - Adjusted bounding box heights based on observed MSL and calculated AGL altitudes
 - Maximum MSL altitude of 12,500 feet (*hard ceiling*)
 - Target maximum AGL altitude of 5,100 feet (*soft ceiling*)
- Consists of 190+ days corresponding to first 14 days of each month from January 2019 through February 2020
 - Initial USA travel ban due to COVID-19 took effect on February 2, 2020 (China only)
 - Schengen Area travel ban took effect on March 13, 2020

695 USA-Based Bounding Boxes
 ≥ 8 nm away from aerodromes in Class B, C, D





Distribution of File Sizes

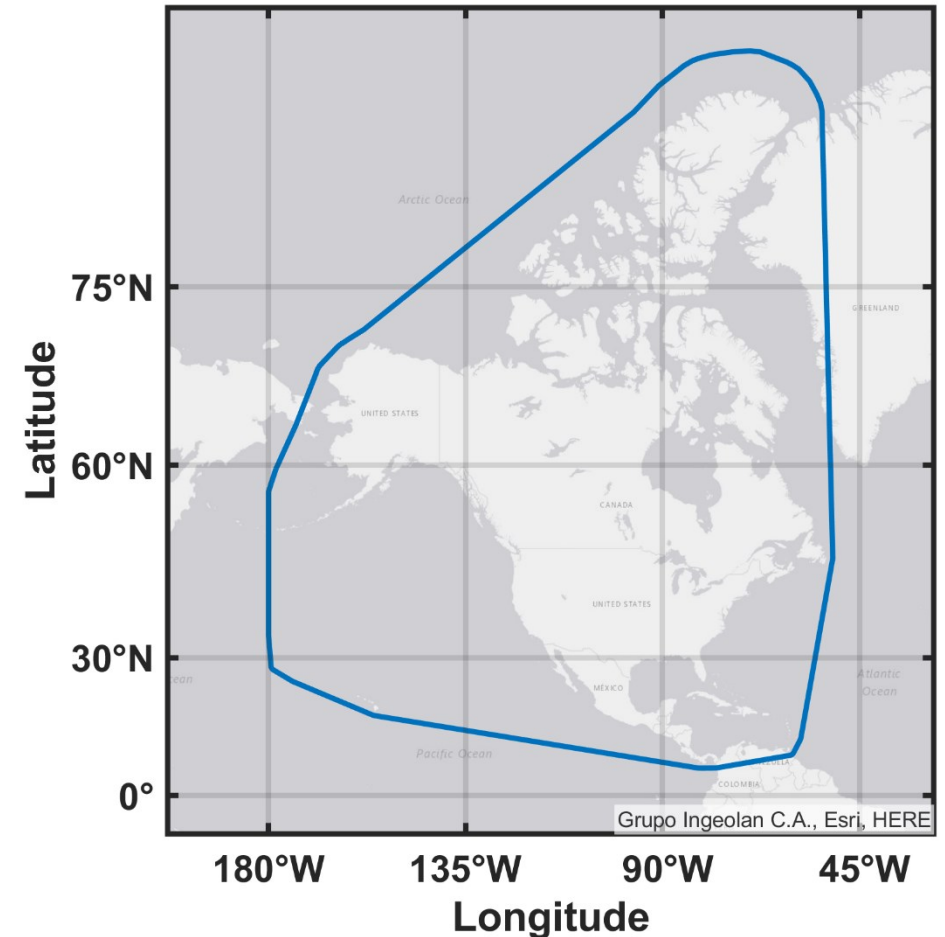




Organizing the Raw Datasets

- **Organize data based on year, aircraft type, and aircraft address (ICAO 24-bit)**
- **Identify aircraft type using the ICAO 24-bit address**
 - Use registries from multiple civil aviation authorities*†
 - Leverage registries from multiple years
- **Filter criteria based on quality and location**
 - Remove observations with incomplete or missing time, latitude, longitude, or altitude information
 - Remove observations outside of user-defined polygon based on ISO 3166-1 A2 boundaries†
- **Convert to U.S. aviation units, as prescribed by the community development contributing guidelines**

Default Geospatial Filtering Polygon





Archiving and Processing

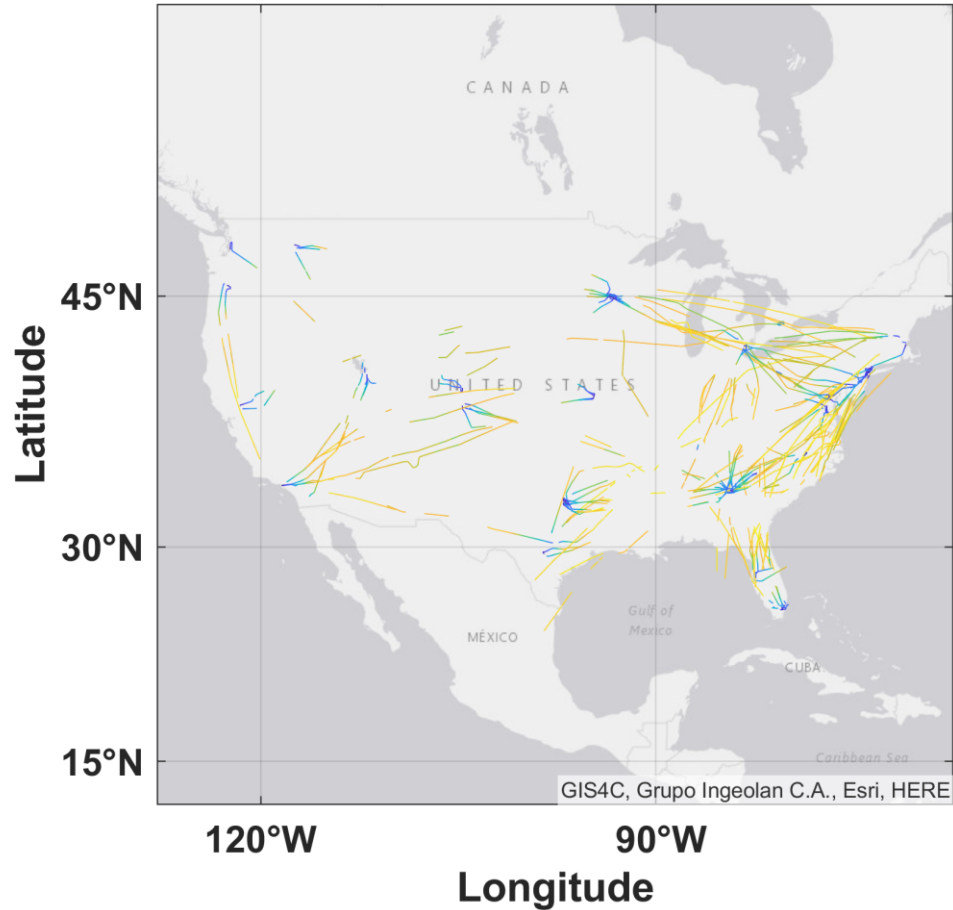
After organizing raw data, it is archived and then processed to generate clean aircraft tracks

- **Organizing raw data can result in tens of millions of files of 1 megabyte or less**
 - **Problematic because small files use a single serialized object storage target**
 - **Inefficient block storage leads to larger random I/O patterns for file access**
- **After organization, data is archived based on ICAO 24-bit aircraft addresses**
 - **After archiving, original inefficient small files can be removed from storage**
 - **Archiving completed through a parallelized process**
- **Archived data is processed for quality control and enhanced with metadata**
 - **Aircraft tracks are segmented, have outliers removed, and interpolated to consistent timestep**
 - **Geospatial data incorporated to estimate AGL altitude, location, etc.**

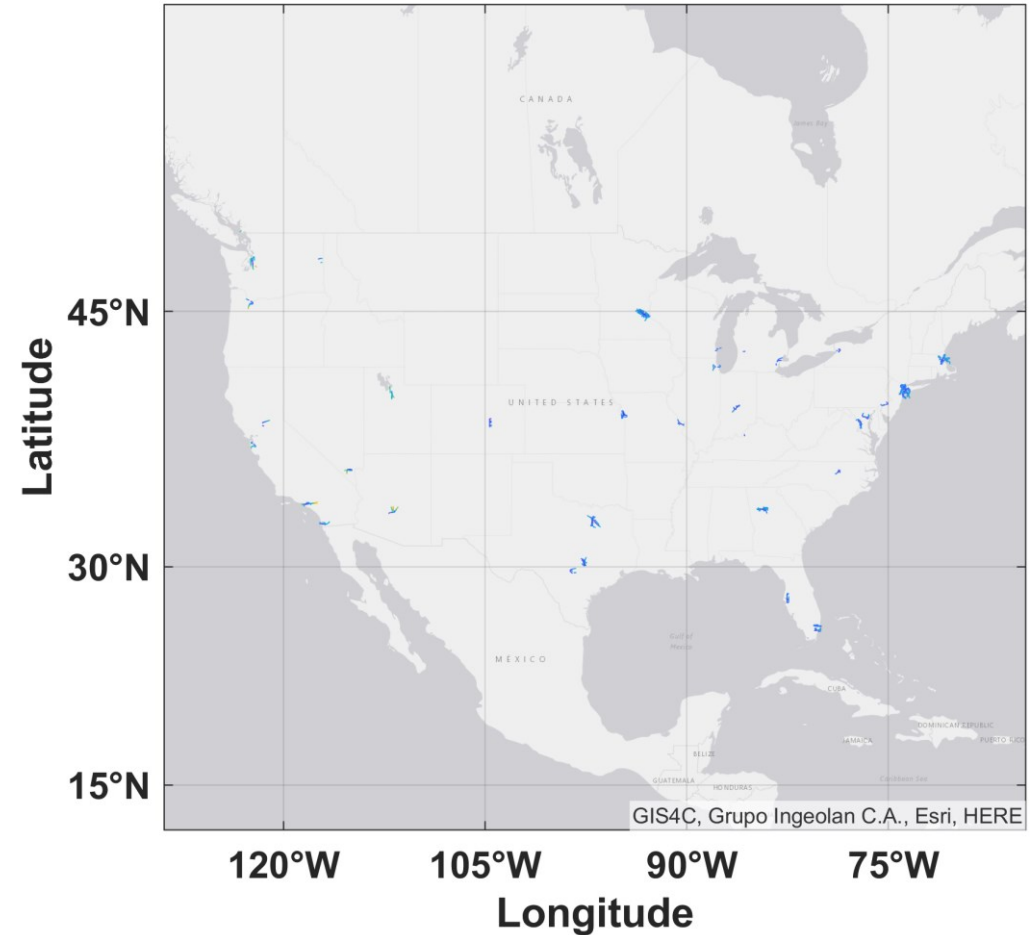


Example Processed Tracks of Fixed-Wing Multi-Engine

Dataset #1: Mondays

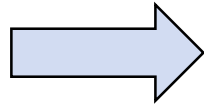


Dataset #2: Aerodromes





- **Introduction**
- **Data Source**
- **High Performance Computing Resources**
- **Benchmark Results**
- **Other Improvements**
- **Conclusion**





MIT Lincoln Laboratory Supercomputing Center (LLSC)


- LLSC has supported critical aviation research for over a decade
- Intel Xeon Phi 64-core nodes* (xeon64c), each of which has 64 compute cores in a single processor socket laid out in a mesh configuration†
 - Connected by a non-blocking 10-Gigabit Ethernet network and a non-blocking Intel OmniPath low-latency network

	TX-Green Upgrade
Processor	Intel Xeon 64 core
Total Cores	41,472
Peak Petaflops	1.724
Top500 Petaflops	1.025 (measured)
Total Terabytes	124
Network Link	Intel OmniPath 25 GB/s



**Based on Nov 2016
Top500.org list**

- #1 in New England
- #2 in the Northeast
- #3 at a US University
- #3 at a University in the Western Hemisphere
- #43 in the United States
- #106 in the World



Only zero carbon emission system in Top500



LLSC Triple-Mode Overview

- **The triple-mode* job launch is a unique job launch mechanism developed at LLSC which provides users with more flexibility to manage memory and threads**
 - **Special array job using a node based scheduler**
 - **Consolidates all the compute tasks running on the same node in a single execution script**
- **Fast resource allocation and job execution by aggregating compute tasks to be executed on the same node as a single scheduling task in an array job**
 - **Enables better performance and more flexibility to manage memory and threads**
 - **Implements explicit process placement and affinity control (EPPAC)**
 - **Assigned exclusive use of each of the requested compute nodes (*LLSC exclusive mode*)**
- **Configured by three parameters when launching job**
 - **Number of compute nodes**
 - **Number of processes per node (NPPN)**
 - **Number of threads per process**



Triple-Mode Parameters

- **LLSC exclusive mode allocates cores by calculating (nodes × slots-per-node)**
 - Number of compute nodes (the first parameter of triples-mode)
 - Slots-per-node is based upon the CPU (xeon64c nodes have 64 slots per node)
- **Routine TX-Green jobs were limited to 4096* xeon64c cores**
 - Due to exclusive mode and allocation limits, maximum number of nodes was 64 nodes
 - 64 nodes × 64 slots per node = 4096 cores
- **LLSC team recommended NPPN ≤ 32 due to memory constraints on xeon64c nodes**
- **Fixed number of threads per process to 1 thread for all benchmarking tests**

Triple-Mode Parameter	Maximum Permitted	Experimental Values
Number of compute nodes	64	64, 32, 16, 8
NPPN	32	32, 16, 8
Number of threads	-	1



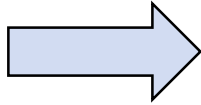
Test Matrix and Allocated Cores

- In addition to triple-mode parameters, need to consider the per-slot memory limit
 - Xeon64c nodes allocated 3 GB per slot on TX-Green
 - Due to potentially large files, requested 2 slots per job for 6 GB per slot memory limit
 - Maximum permitted compute cores dependent upon slots per job
 - Slots per job is different than slots per node when using exclusive mode

Nodes	NPPN	Threads	Compute Cores	Slots Per Job	Permitted Compute Cores	Slots Per Node	Allocated Cores
64	32	1	2048	2	2048	64	4096
32	32	1	1024	2	2048	64	2048
16	32	1	512	2	2048	64	1024
8	32	1	256	2	2048	64	512
64	16	1	1024	2	2048	64	4096
32	16	1	512	2	2048	64	2048
32	8	1	256	2	2048	64	2048
64	8	1	512	2	2048	64	4096



- **Introduction**
- **Data Source**
- **High Performance Computing Resources**
- **Benchmark Results**
- **Other Improvements**
- **Conclusion**





Benchmarking Organization of Dataset #1: Mondays

- **Benchmarked organization of 2425 files (714 Gigabytes) from dataset #1**
 - Maximum possible number of CPUs was 2048, this is a good dataset to assess job overhead
 - Wide time span of data and wide distribution of file sizes to assess task organization
 - Presented analysis focuses on dataset #1, but similar results were observed with dataset #2
- **Tasks organized either chronologically or by size**
 - Chronological organization has the earliest date first and the most recent date last
 - Size organization has the largest file first and the smallest file last
- **Tasks allocated to workers either in a batch or via self-scheduling**
 - Batch workflow maps the tasks across workers at the start of the job
 - Self-scheduling workflow consists of two roles: 1 × manager, N × workers
- **Benchmark results organized given nodes, NPPN, and data organization**
 - Total job time to complete all tasks, as measured by the manager
 - Distribution of time by workers

Metrics



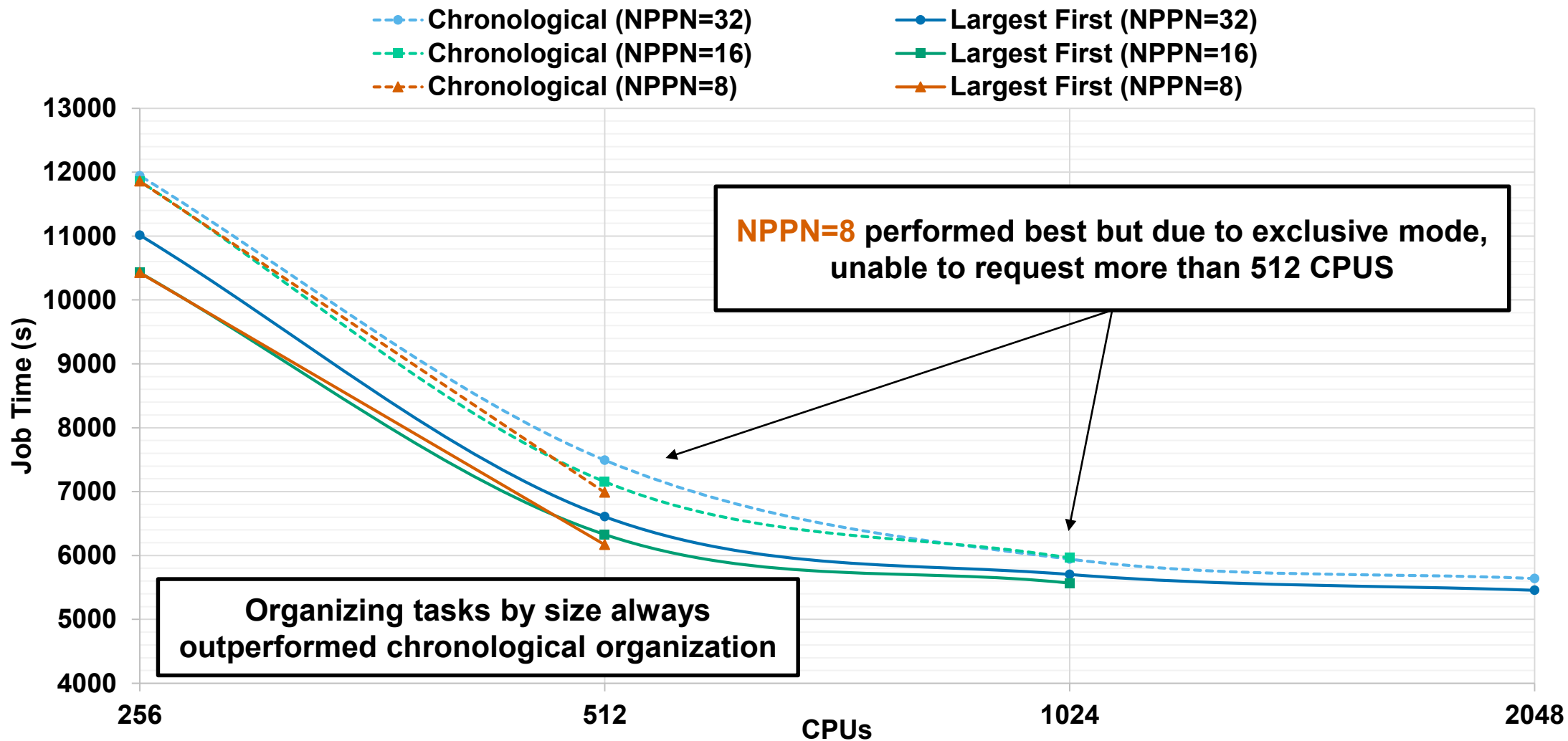
Self-Scheduling Workflow: One Manager, Many Workers

- 1. Manager sends initial set of tasks to all workers in sequential order**
 - Task 1 sent to Worker 1, Task 2 sent to Worker 2, etc.
 - Tasks allocated as fast as possible, manager does not pause when sending messages
- 2. Workers receive and complete initial tasks, and then reports back to manager**
 - Workers only assigned and complete one task at a time
 - Worker waits 0.3 seconds prior to checking for another message from the manager
 - Manager sequentially receives messages and counts how many total tasks are completed
- 3. If not all tasks are completed, manager sequentially sends tasks to idle workers**
 - Manager waits 0.3 seconds prior to checking for more idle workers
 - Rate at which workers complete tasks and become idle depends task organization
 - If many workers are idle, workers may wait minutes prior to receiving next task
- 4. Steps 2 and 3 repeated until all tasks completed, job is then terminated**



Benchmarking Job Time to Organize Dataset #1

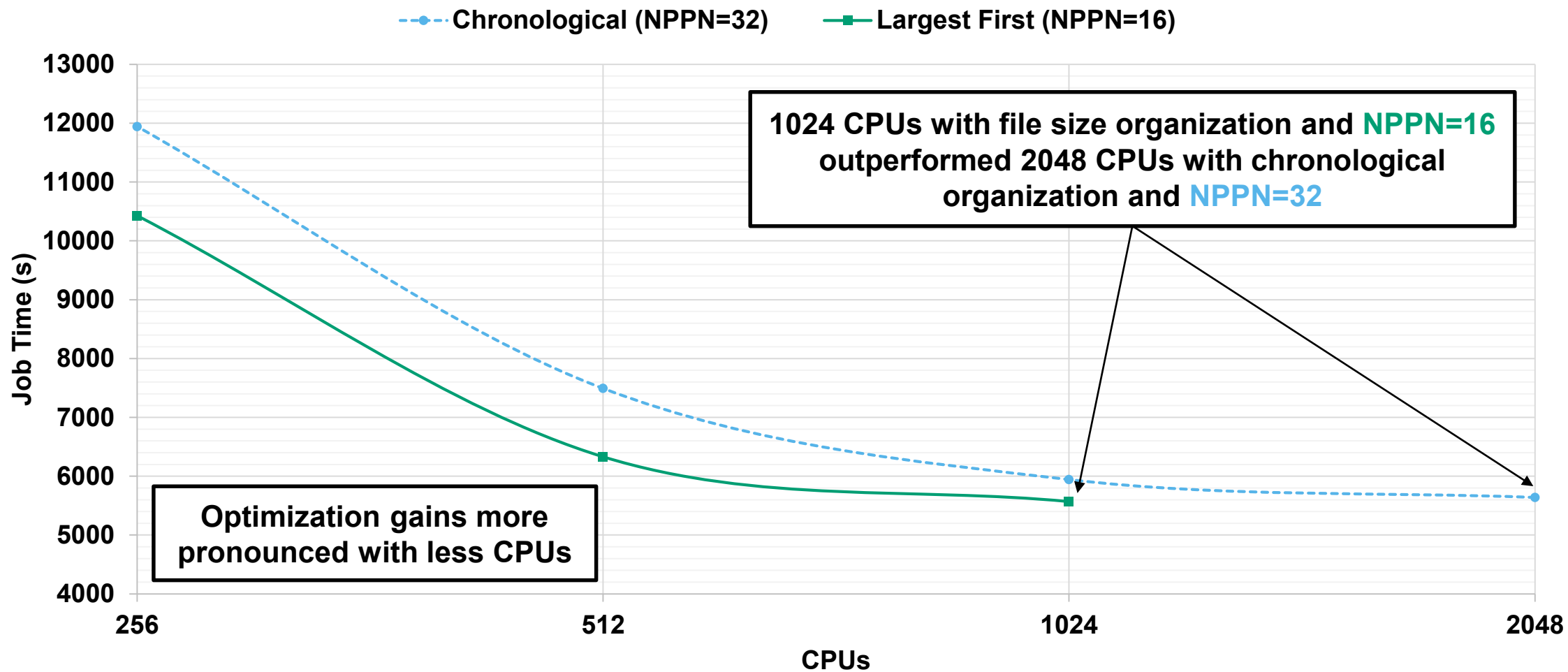
Self-Scheduling





More CPUs are not necessarily faster

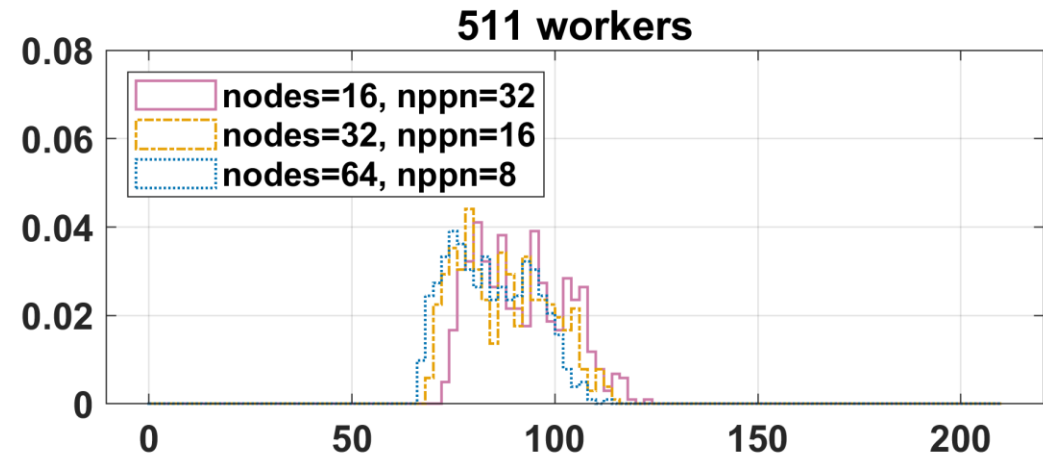
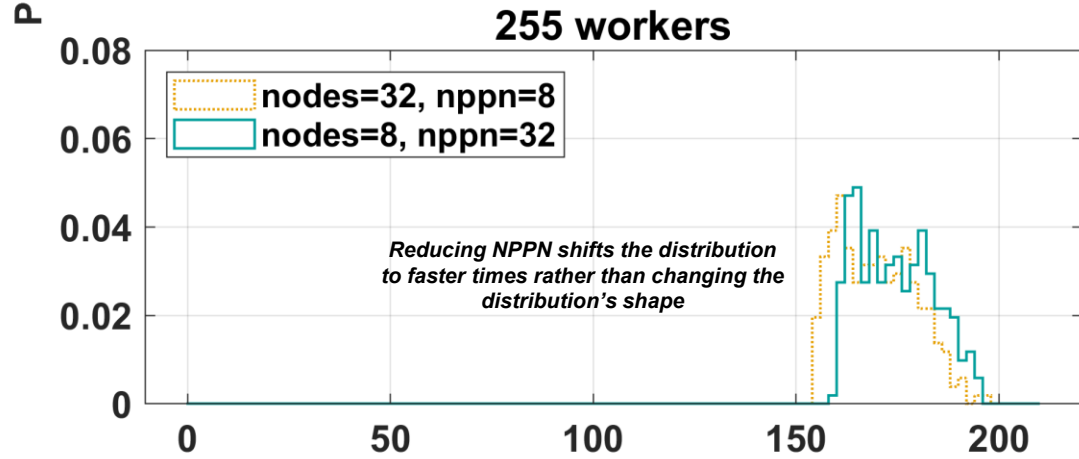
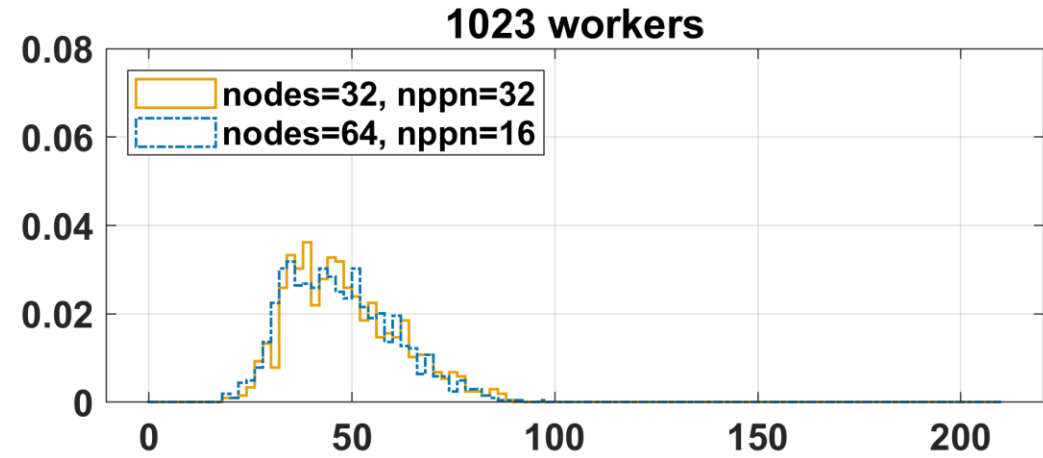
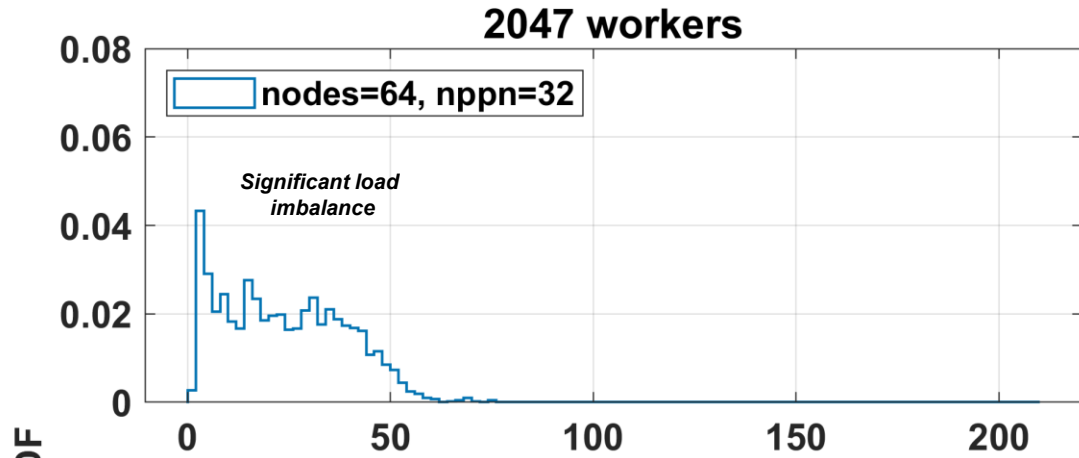
Self-Scheduling





Worker Time: Tasks Organized Chronologically

Self-Scheduling

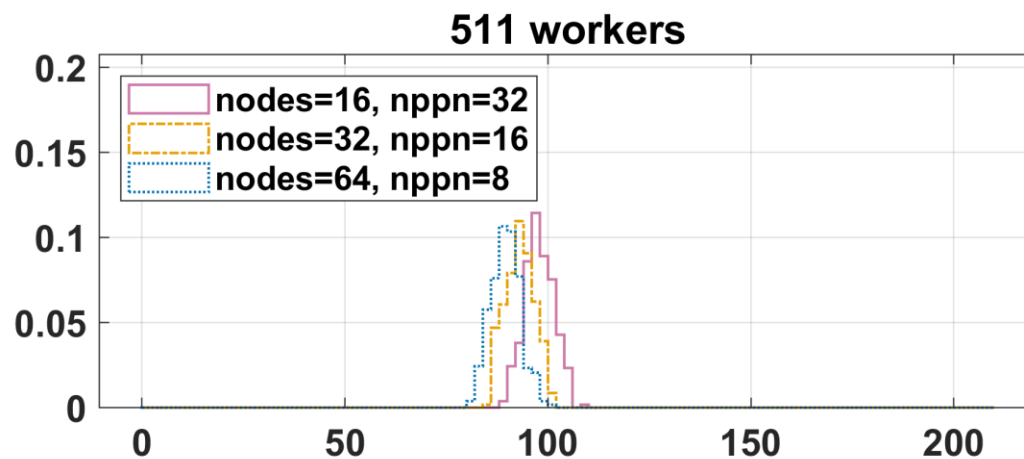
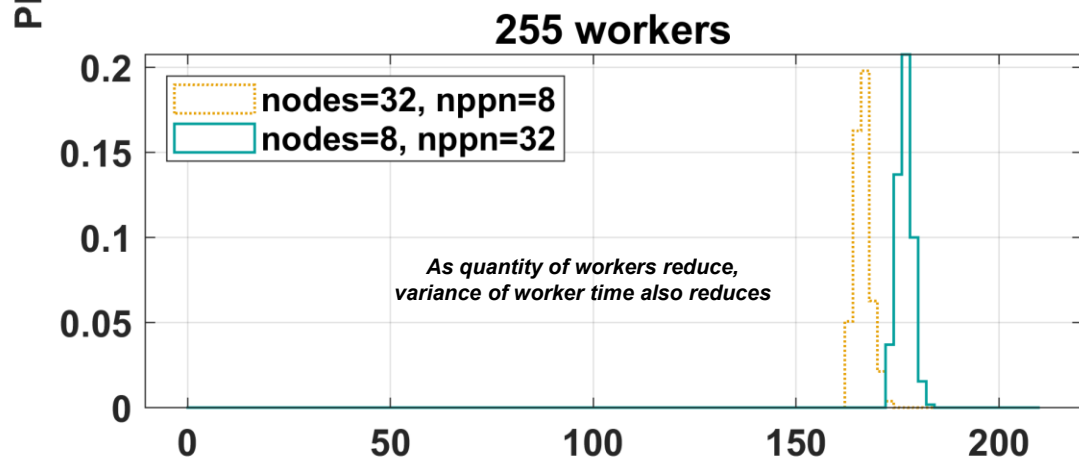
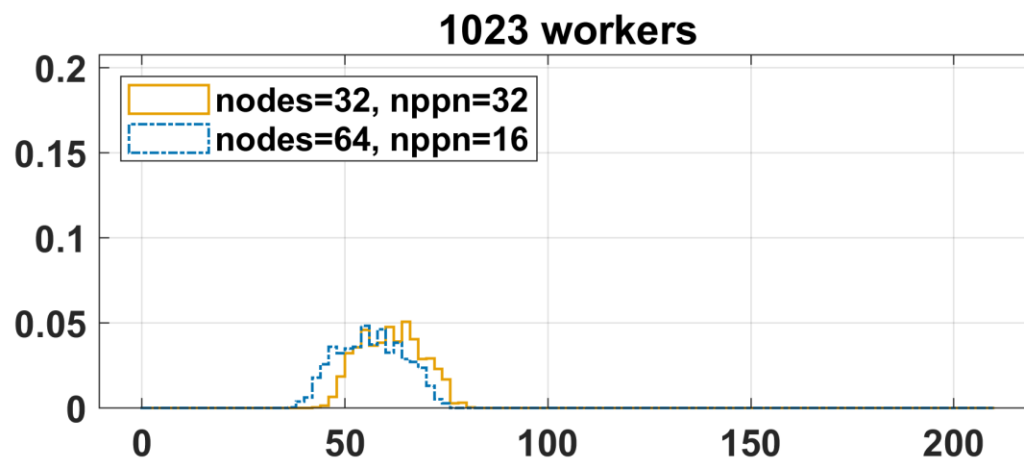
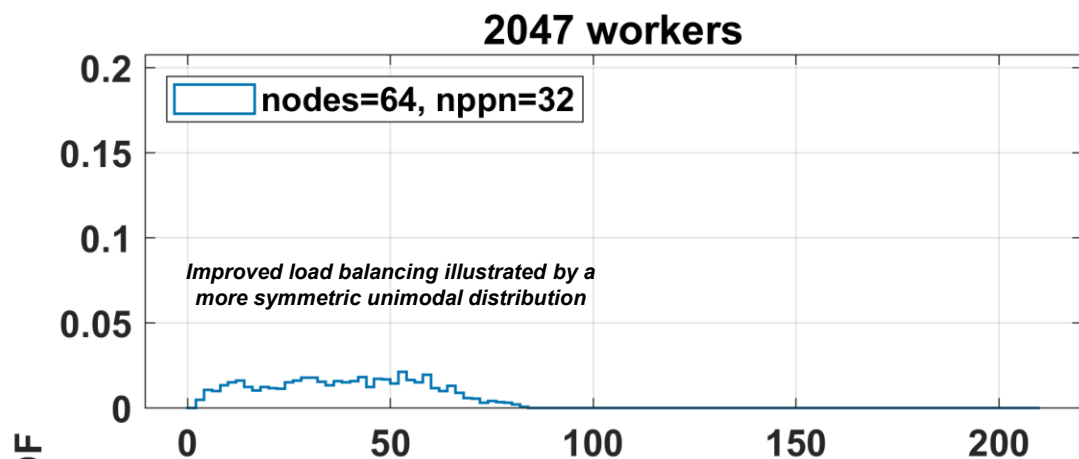


Worker time (minutes)



Worker Time: Tasks Organized by Size

Self-Scheduling



Worker time (minutes)



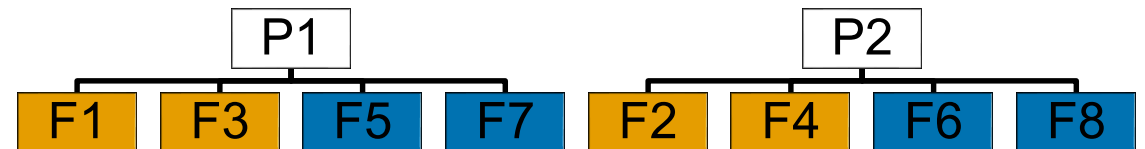
Benchmarking Batch Workflow

- **Batch workflow benchmarked after assessing self-scheduling workflow**
 - Better performance using file size organization and reducing NPPN already identified
 - Self-scheduling overhead also identified as a potential issue
 - Based on lessons learned, limited benchmarking to Nodes = 64, NPPN = 8, Threads = 1
- **Batch workflow performance allocates all tasks upfront, with performance significantly dependent on how tasks are distributed**
- **Tasks allocated via block or cyclic distribution**

Block Distribution
Distributed sequentially



Cyclic Distribution
Distributed round robin (*Like dealing cards*)





Batch Workflow

- **Based on the size of the input dataset, the total number of tasks are calculated**
 - Dataset #1 has 2425 files, so there are 2425 tasks
 - Quantity of tasks independent of how tasks are prioritized (chronological, size, etc.)
- **Tasks sorted based on distribution**
 - Block distribution sorts tasks chronologically (otherwise worker #1 assigned the largest files)
 - Cyclic distribution sorts tasks by file size
- **A mapper function allocates a batch of tasks to each worker**
 - Batch of tasks are issued upfront, worker knows all assigned tasks at start of job
 - Each batch has a similar quantity of tasks, with workers never idle
- **No communication between workers, job ends when last worker finishes**

When sorting tasks by file size, lower rank workers (i.e. 1, 2, ...) will be, on average, tasked to process more data, which should result in longer compute time than higher rank workers



Summary Statistics and Discussion: 512 CPUs

Self-Scheduling Reduces the Span of Worker Time

Worker Time (seconds)

Workflow	Organization	Job Time	Minimum	Mean	Median	Maximum	Span
Batch - Cyclic	File size	10225	2203	5839	5804	10225	8022
Self-scheduling	Chronological	6989 - 7493	4011 - 4367	5062 - 5506	5001 - 5435	6790 - 7354	2779 - 2987
Self-scheduling	File size	6171 - 6608	4860 - 5365	5378 - 5872	5381 - 5863	6042 - 6521	1156 - 1182

- **Self-scheduling workflow drastically reduced job time due to improved load balancing**
 - Message overhead between manager and workers was non-trivial
 - Future work should focus on reducing idle time of workers
- **File size-based task organization performed better**
 - When self-scheduling, 11% difference of maximum worker time between task organizations
 - Minimizing NPPN was preferable, but not the significant driver for performance



- **Introduction**
- **Data Source**
- **High Performance Computing Resources**
- **Benchmark Results**
- **Other Improvements**
- **Conclusion**





Download, Archiving, and Processing Dataset #2

- **Benchmarked download, archiving, and processing of dataset #2**
 - Applied lessons learned from benchmarking the organization step
 - Updated workflow to leverage triples-mode, self-scheduling, and cyclic distribution
- **Downloading involves querying the OpenSky Impala database**
- **Archiving consists of zipping bottom level directories created during organization**
- **Processing consists of cleaning, interpolating, and enhancing archived data**
 - The processing output is the input to the model training
 - Includes multiple geospatial calculations that can be computationally intensive



Dataset #2: TX-E1 to TX-Green

- **136,884 queries executed on OpenSky Impala database to generate dataset #2**
 - Serial queries completed over 3-5 SuperCloud (TX-E1) CPUs required about a month to finish
 - Queries recorded in .txt format and format to .csv via parallelization with LLMapReduce*
- **TX-E1 login nodes have open access to the internet, TX-Green login nodes do not**
 - SuperCloud enabled us to immediately start downloading open source data
 - Completing the same task with LLSC would have required a writing and getting approval for a special firewall exemption; this overhead likely would have taken at least a month
- **Data moved using rysnc via special high bandwidth link between TX-E1 and TX-Green**
 - Transfer required 5-6 hours, with observed speeds of at least 33 MB/s (based on rysnc -p)
 - Did not compress or archive files prior to transfer
 - Hypothesize that if files were archived, speeds would have probably been closer to 100 MB/s, but tradeoff to spend time archiving wasn't practical



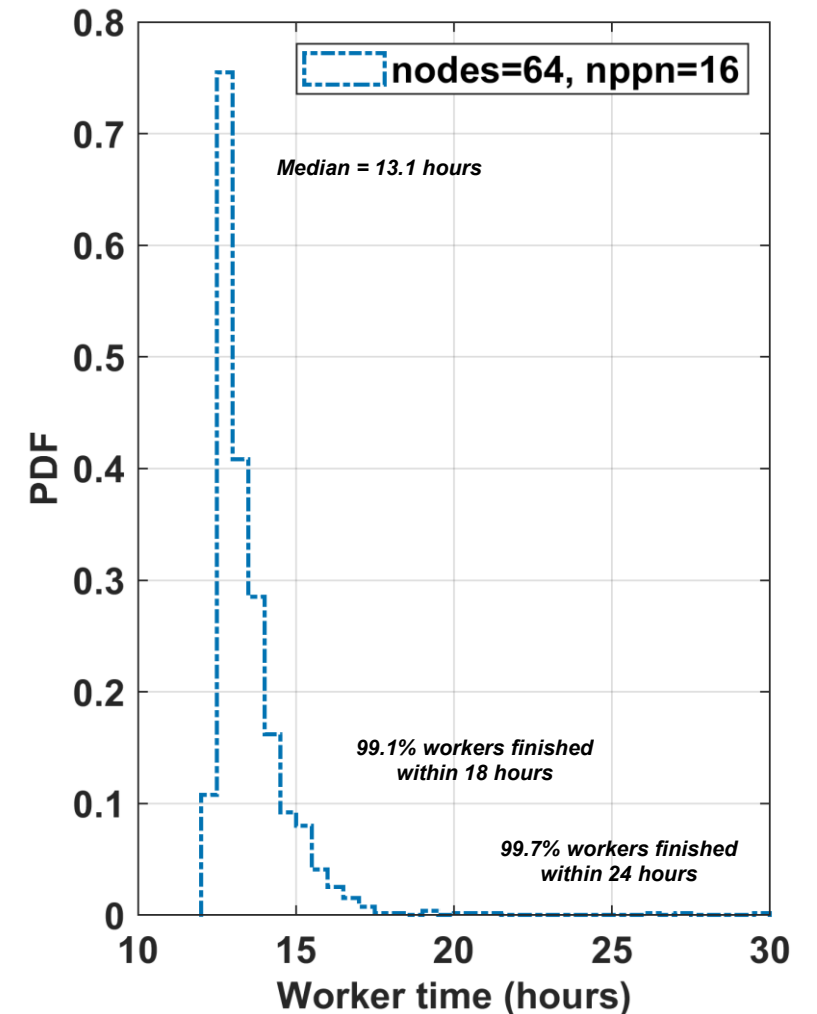
Data Archiving Using Cyclic Distribution

- **After organizing the data, it is archived for efficient HPC storage***
 - Parallelized task using LLMapReduce, with tasks organized into set of directories to archive
 - Depending on dataset, millions of directories needed to be considered
- **Originally block distributed data to parallelize processes**
 - Block distribution resulted in poor load balancing
 - 2% of processes accounted for 95%+ of the required time
- **90%+ decrease in job time by changing to cyclic from block distribution (*days to hours*)**
 - LLMapReduce sorts input based on filename, which differs how cyclic distribution was implemented for organizing the datasets using pMATLAB
 - Similar speed improvement for both datasets



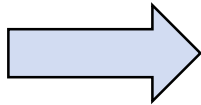
Process and Interpolate

- **Benchmarked processing of organized and archived dataset #2 on TX-Green**
 - Single benchmark with 1024 CPUs with self-scheduling and triples-mode
 - 23,644 unique aircraft randomly ordered into tasks
- **Job required 29.6 hours to complete**
 - Batch allocation required at least 7 days by comparison
 - Speed improvement enables more rapid future analyses
- **Load balancing issues mitigated but still remain**
 - 17.3 hours difference between fastest and slowest workers
 - 16.4 hours difference between slowest worker and median
- **Future work required to improve load balancing**





- **Introduction**
- **Data Source**
- **High Performance Computing Resources**
- **Benchmark Results**
- **Other Improvements**
- **Conclusion**





Summary

- **To support development and evaluation of aircraft safety of life system, aircraft behavior often represented by statistical encounter models**
- **Growth in complexity of analyses has required growth in model sophistication and high performance resources**
- **Improved efficiency of previously developed processing workflow to enable more rapid and comprehensive analyses**
- **Achieved substantial decrease in time to completed jobs through benchmarking, intelligent job launching, and improved load balancing**
 - **Self-scheduling**
 - **Triples mode**
 - **Cyclic distribution**

Time to fully execute workflow reduced from weeks to days



Thank You!



Questions?

Feedback?

Presenter: Andrew Weinert

Contributors (alphabetical): Marc Brittain, Matthew Edwards, Randal Guendel, Christine Serres, Ngaire Underhill

Homeland Protection and Air Traffic Control Division

Email: andrew.weinert@ll.mit.edu



New Airspace Entrant Opportunities



Space Launch/Reentry



Very High Altitude Loiter



Large UAS missions



Mid-sized UAS operations



Ubiquitous Small UAS



Urban air mobility

Growing demand to enable wide range of new vehicle and mission types



Processor Types on LLSC TX-Green



Node Types	AMD Bulldozer	Intel Broadwell	Intel Broadwell – High Memory	NVIDIA K80 on Broadwell – High Memory	Intel Knights Landing (KNL)	Intel Cascade Lake	NVIDIA Volta V100 on Cascade Lake
Number of Nodes	274	340	35	70 GPUs	632	224	448 GPU
Processors per Node	2x Opteron 5274	2x Xeon E5-2683 v3	2x Xeon E5-2680 v4	2x K80	1x Xeon Phi 7210	2x Xeon Gold	2x V100
Sockets / Cores per Socket / Total Cores per Node	2 / 16 / 32	2 / 14 / 28	2 / 14 / 28	2 / 2x 78 SMs / 156 SMs	1 / 64 / 256 AVX-512 units	2 / 20 / 40	2 / 2x 80 SMs / 160 SMs
Clock Speed	2.2 GHz	2.0 GHz	2.4 GHz	562/875 MHz	1.3 GHz	2.5 GHz	1290/1455 MHz
Main Memory	128 GB	256 GB	512 GB	24 GB per GPU	256 GB	384 GB	32 GB per GPU
Network(s)	10-GigE	10-GigE	10-GigE	–	10-GigE	10-GigE	–
Local Storage	9.0 TB	9.0 TB	1.0 TB	–	1.0 TB	4.0 TB	–



Green AI Accelerator: World Leader in Interactive AI Supercomputing



Low Carbon Emission

- Significant increase in computing power for simulation, data analysis, and machine learning
- Leverages power of 900 Nvidia Volta GPUs



- Largest AI Research System at a University

	Capability
Processor	Intel Xeon & Nvidia Volta
Total Cores	737,000
Peak	7.4 Petaflops
Top500	5.2 Petaflops (#42 in World*)
Memory	172 Terabytes
Peak AI Flops	100+ Petaflops (#6 in World*)
Network Link	Intel OmniPath 25 GB/s



Increasing Tasks Per Message Degraded Performance

- The manager can send multiple tasks per message to workers
- Briefly investigated if increasing the tasks per message improved performance
 - Only tested with one triples mode configuration*
 - Limited to Dataset #1: Mondays
- Tasks allocated via cyclic distribution
 - Modified self-scheduling task prioritization
 - Assumed that block (sequential) distribution would had increased computational burden on low rank workers
- Performance decreased as tasks per message increase
- Further investigation required

