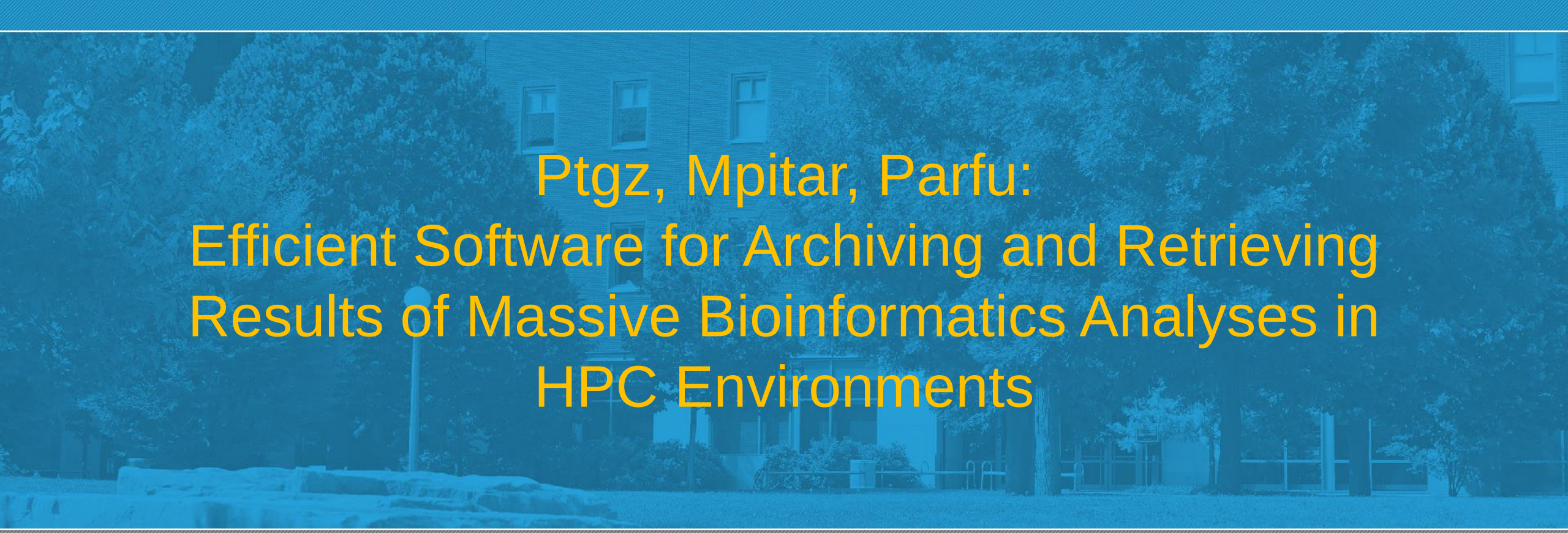
Ptgz, Mpitar, Parfu: Efficient Software for Archiving and Retrieving Results of Massive Bioinformatics Analyses in HPC Environments

Craig Steffen, NCSA

HPC User Software and Tools Workshop

November 19, 2021



Ptgz, Mpitar, Parfu: Efficient Software for Archiving and Retrieving Results of Massive Bioinformatics Analyses in HPC Environments

RYAN CHUI, CHRISTINA FLEIGE, ROLAND HAAS,
KATHERINE KENDIG, LIUDMILA MAINZER, **CRAIG P STEFFEN**

HPC USER SOFTWARE AND TOOLS WORKSHOP: NOVEMBER 19, 2021

ILLINOIS
NCSA | National Center for Supercomputing Applications

Authors

- NCSA Genomics Group
  - Ryan Chui*, Christina Fleige, Katherine Kendig, Liudmila Mainzer

- NCSA Blue Waters Science and Engineering Applications Support team (SEAS)
  - Roland Haas, Craig Steffen

* Now at “23 And Me”

Outline

- The problem (lots of files)
- Other potential solutions
- Our codes
- Testing
- Performance
- Conclusions

Background: User Software Of Concern

- Bioinformatics Workflows
- Astronomy Image Processing Workflows

Characteristics in common: produces *numerous* files while running, which have to be managed later

Software Developed Running on Small Systems

- Local File System Write Cache (in RAM)
- Metadata local (in RAM)
- Very fast, efficient with local file systems
- Allows file system to be used ”as database” (one file per entry)
  - *Many many (millions)* small files to minimize access times

Symptom: This Doesn't Scale Well

- HPC parallel file systems assume:
  - Small meta-data load per gigabyte stored (few large files)
  - Files are large (take advantage of distributed byte storage)
  - Directories have small numbers of files (Blue Waters Lustre file systems slow down when running `ls` in a directory with >5000 entries
- Tape systems assume
  - Files can be stored in any arrangement: are stored by convenience of tape systems
  - Small files store very efficiently to disk cache, then get written across dozens or *hundreds* of tapes
  - Retrieval is very slow and sometimes impractically long
- File transfer
  - Globus is configuration limited to a fixed number of files at a time
  - For numerous small transfers, setup latency dominates total transfer time

What Do We (Ultimately) Do About This?

- Change The Workloads
  - (use database/container files like HDF)
  - Bioinformatics is changing very quickly
  - It's a very large and fast-growing field
- Change the File Systems?
  - File systems specs change *slowly*
  - Many small files are not a popular use case

Short-Term Solution: Tar Up Directories

- Shift file logistics from OS to user
- After files written, use bundles files for storage or transfer
- How?
  - Unix Tar (“Tape ARchive”)
  - Tar + gz (Tar + compression)
  - Tar + Pigz (tar + *parallel* compression)
- BUT
  - Single-threaded (pigz multi-threaded, single-node)
  - Too slow
  - For large data collections, tar-ing big directories (hundreds of gigabyte) would exceed max job time, makes a drag on workflow

What We Needed:

- A *parallel, scalable* version of tar
- One did not exist in 2014/15 when we started working on these three packages independently

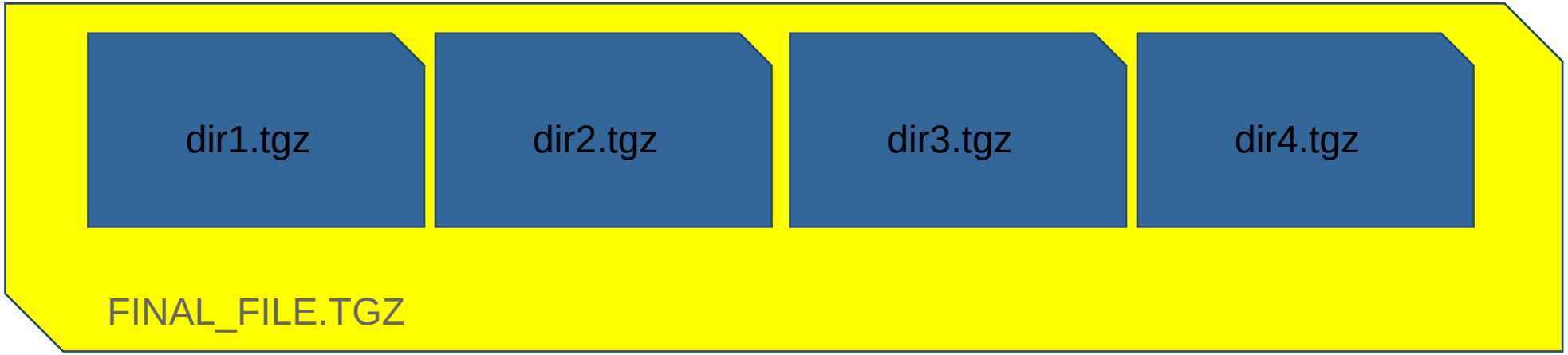
Solutions We Didn't Test Against

- Pltar
  - No longer under development (confirmed with ORNL)
- Fcp
  - File copying/moving, not file packaging
- Htar
  - Tied to specific HPSS tape installations
  - Not a general tool
- Dtar
  - Part of MpiFileUtils from LLNL
  - Released in non-test form fall of 2021: too recently for comparison
  - Definitely needs comparison in the future

Featured Solution #1: ptgz

- The NCSA Genomics group developed ptgz
  - Uses multiple threads across multiple nodes of standard Unix tar and gzip
  - Creates many .tar.gz (or .tgz) intermediate archive files on disk
  - Intermediate archive files then archived into a final single archive file
  - Outputs an external separate index file (list of all files encapsulated in the archive)
  - Written in C++

Ptgz File Format

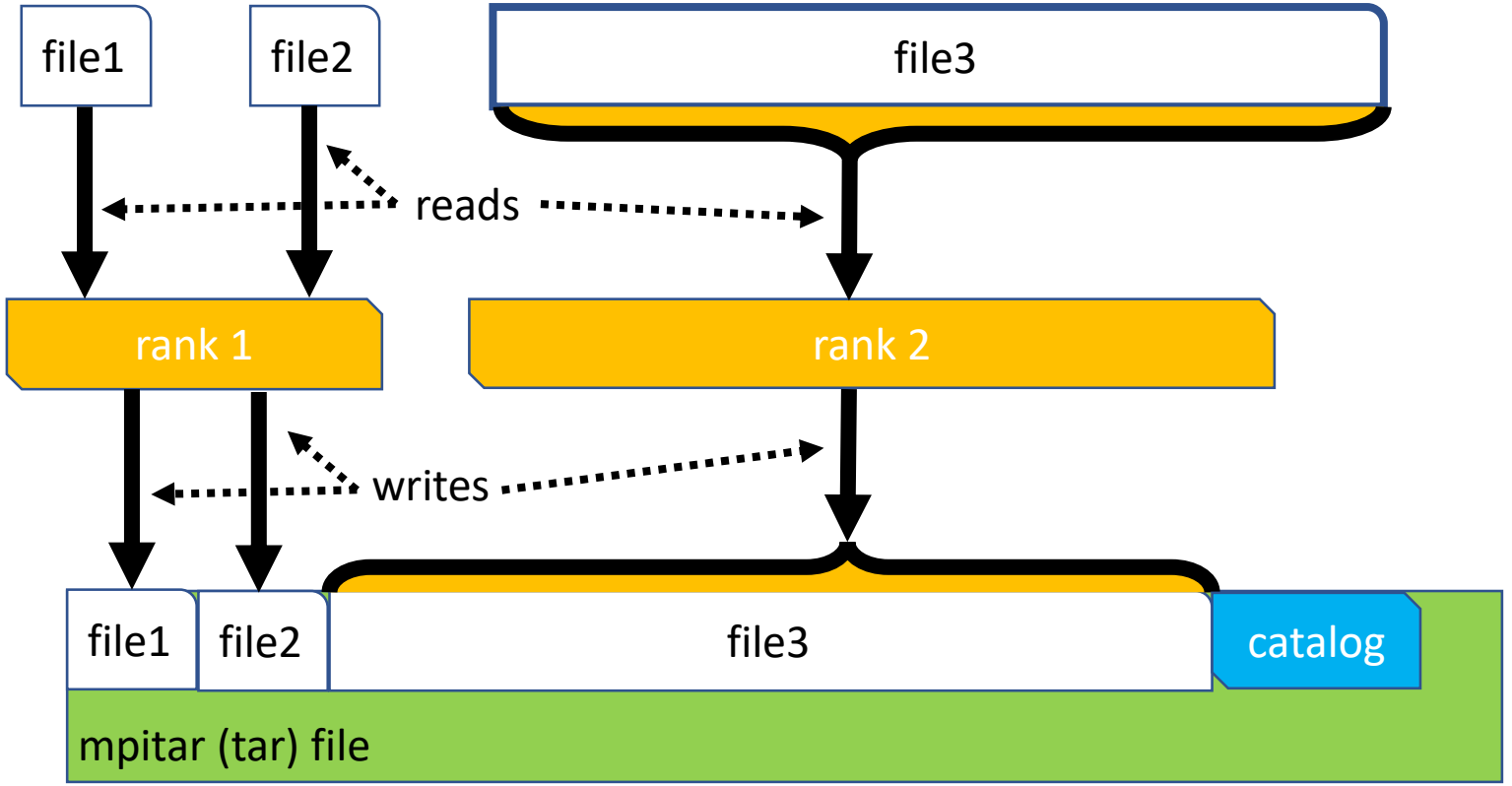


Index file
File1.dat
File2.txt
File3.c

Featured Solution #2: mpitar

- NCSA SEAS team developed mpitar
- MPI parallel application that scales across many nodes
- Implements tar file format directly
- Rank 0 scans directories, archiving begins as soon as first directory listing is complete
- File format: Tar file with all target files, embedded index file at the end
- Written in C++

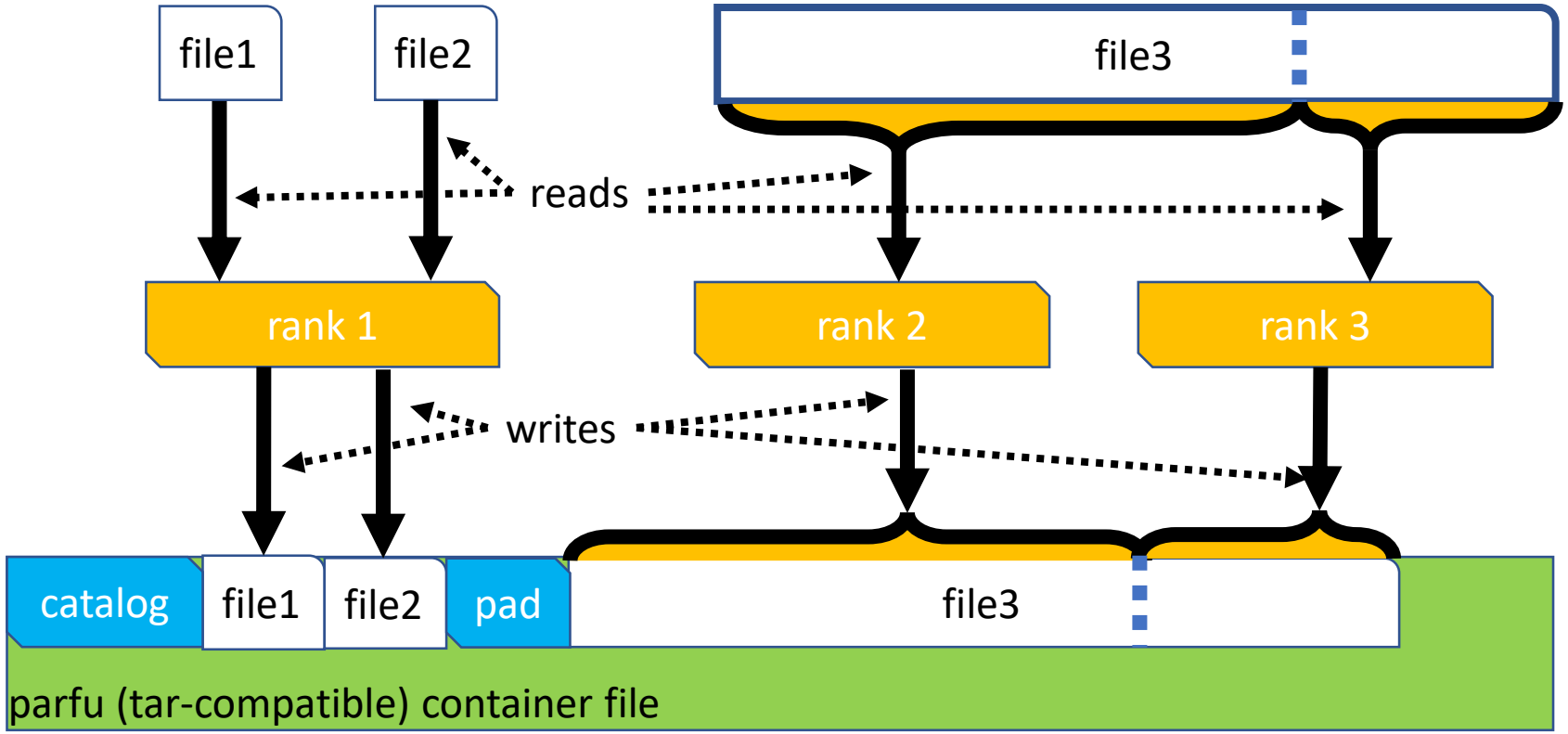
Mpitar file format



Featured Solution #3: parfu

- NCSA SEAS team developed parfu
- Originally implemented proprietary format, now uses tar standard files
  - File is tar standard file format, with *first* file an index of archive contents
- (current version) all directory scanning by thread zero, all directory scanning finished before archiving begins
- Written in C and C++
- Being re-written into C++ (winter 2021-2022)

Parfu file format



Performance Testing

- Tested ptgz, mpitar, and parfu on NCSA (Industry Group) iForge cluster to compare to standard tools
- Also tested standard tools (tar, tar+gz, tar+pigz) on iForge
- Tested mpitar and parfu on NCSA Blue Waters and TACC Stampede 2 to characterize scaling behavior at high node count

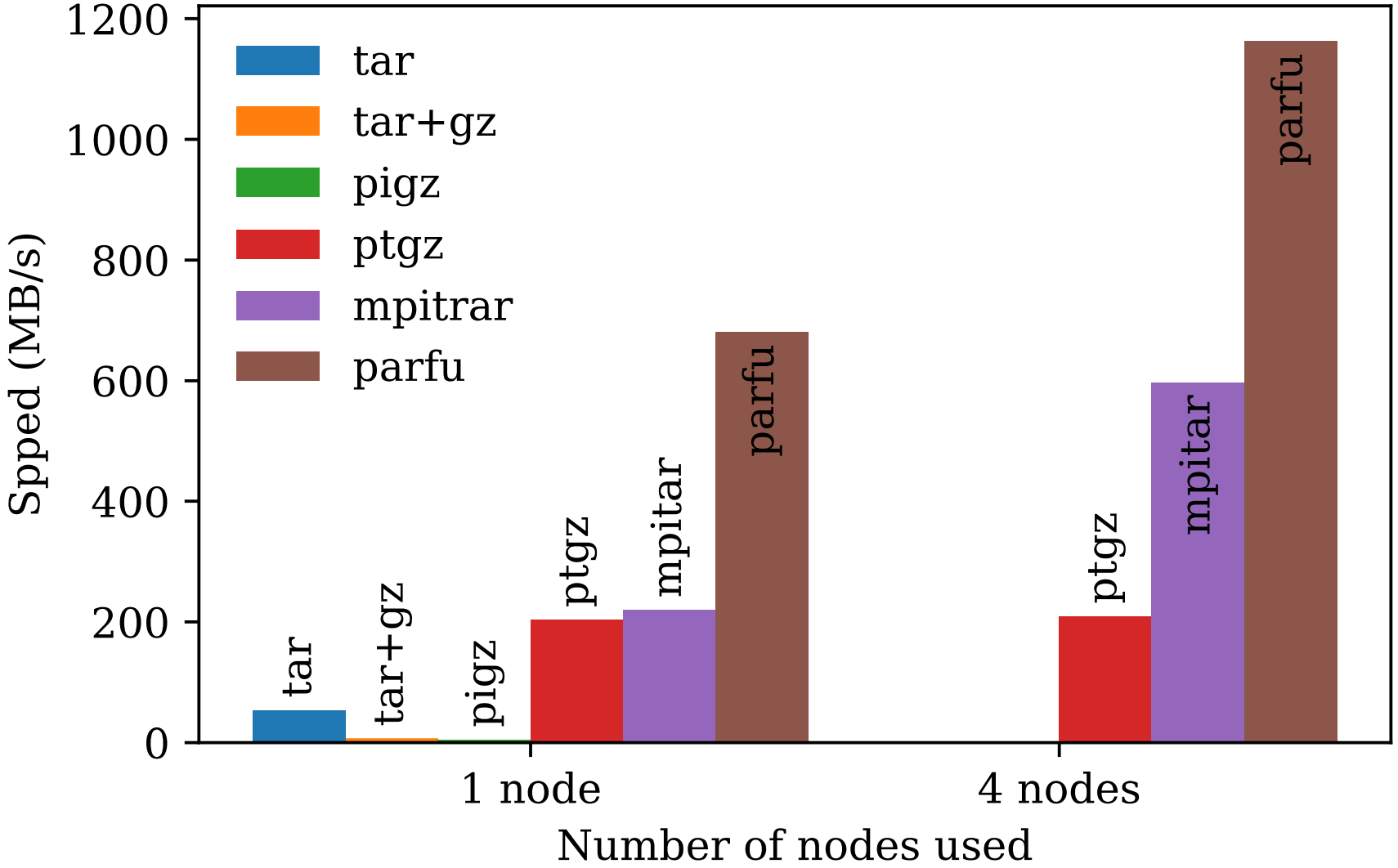
Performance Testing Data Sets

- Genomics group provided three typical bioinformatics data test sets to be used for all performance testing
  - GWAS
    - 198,281 files
    - 27 GB
  - Arabdiopsis
    - 1,423,280 files
    - 492 GB
  - VarCall
    - 2027 files
    - 732 GB

Performance Testing Systems

- iForge
  - NCSA Industry Group
  - Skylake processors 40 cores per node
  - 192 GB RAM per node
  - GPFS file system
- Stampede 2
  - Texas Advanced Computing Center
  - Skylake processors 48 core per node
  - 192 GB RAM per node
  - Lustre file system
- Blue Waters
  - NCSA
  - AMD Interlagos processors, 32 integer/16 FP cores per node
  - 64 GB RAM per node
  - Lustre file system

Performance: GWAS on iForge

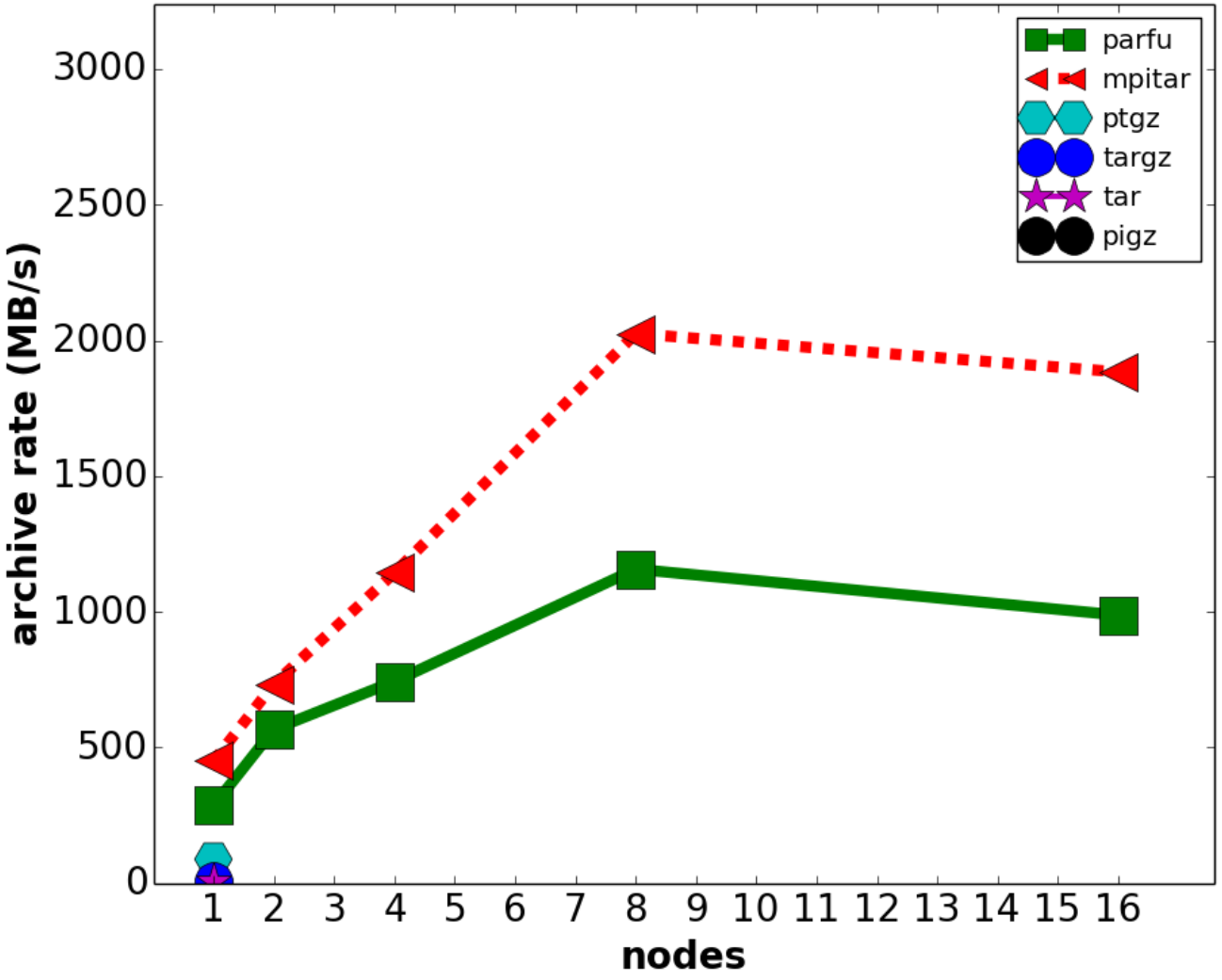


Performance on iForge

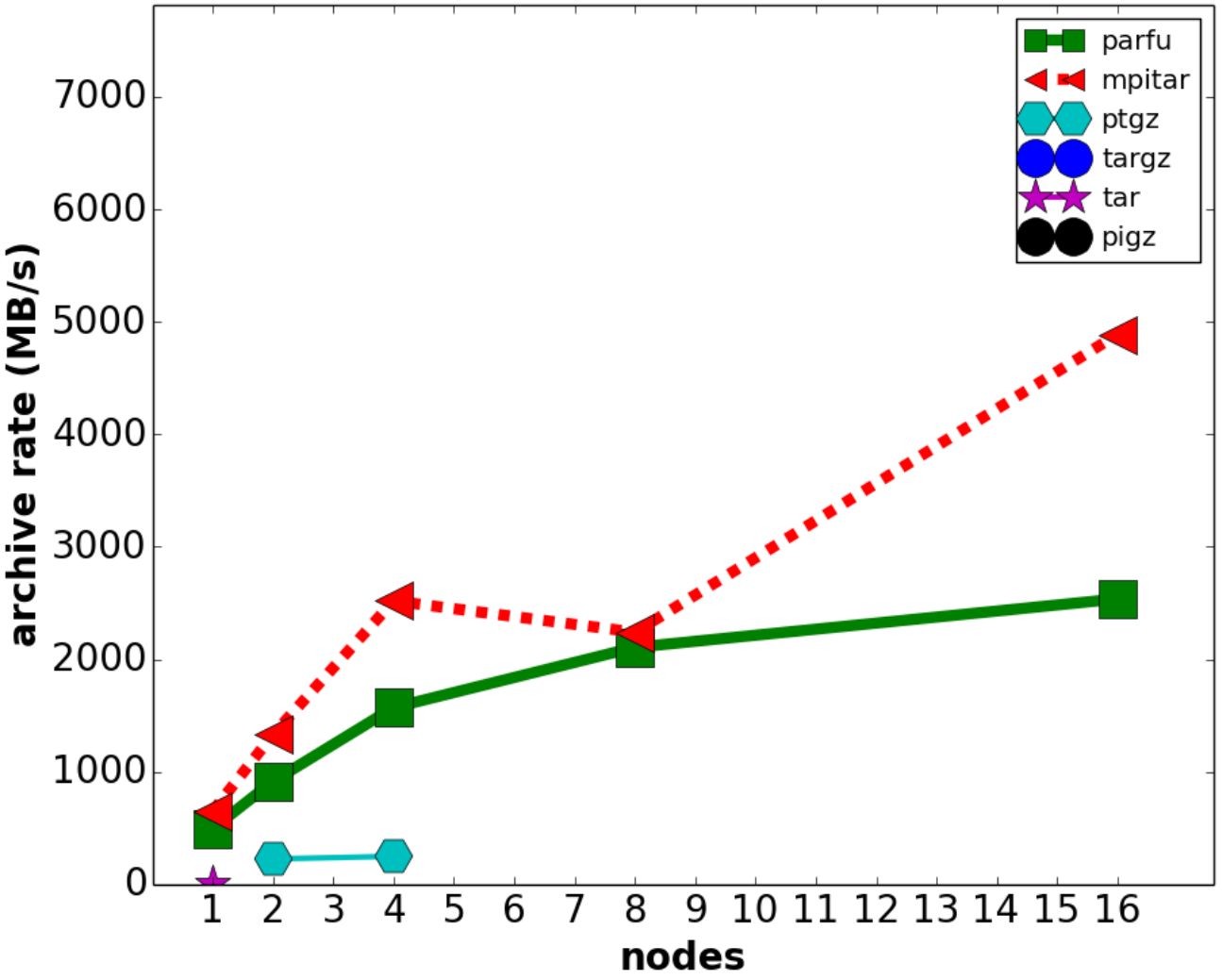
TABLE III
PACKAGING TOOL PERFORMANCE ON IFORGE.

Code	Data	Nodes	Transfer Time (s)	Rate (MB/s)	Rate Per Node (MB/s/node)
tar	GWAS	1	496	54	54
tar+gz	GWAS	1	3756	7	7
pigz	GWAS	1	5499	5	5
ptgz	GWAS	1	132	204	204
ptgz	GWAS	4	128	210	52
mpitar	GWAS	1	122	220	220
mpitar	GWAS	4	45	596	149
parfu	GWAS	1	39	681	681
parfu	GWAS	4	23	1163	291
tar	Arabidopsis	1	19634	25	25
tar+gz	Arabidopsis	1	52825	9	9
ptgz	Arabidopsis	1	1981	248	248
ptgz	Arabidopsis	4	2025	243	61
mpitar	Arabidopsis	1	2483	198	198
mpitar	Arabidopsis	4	639	769	192
parfu	Arabidopsis	1	795	619	619
parfu	Arabidopsis	4	342	1436	359
tar	VarCall	1	2176	336	336
tar+gz	VarCall	1	23448	31	31
mpitar	VarCall	1	2871	255	255
mpitar	VarCall	4	5962	123	31
parfu	VarCall	1	289	2527	2527
parfu	VarCall	4	108	6778	1694

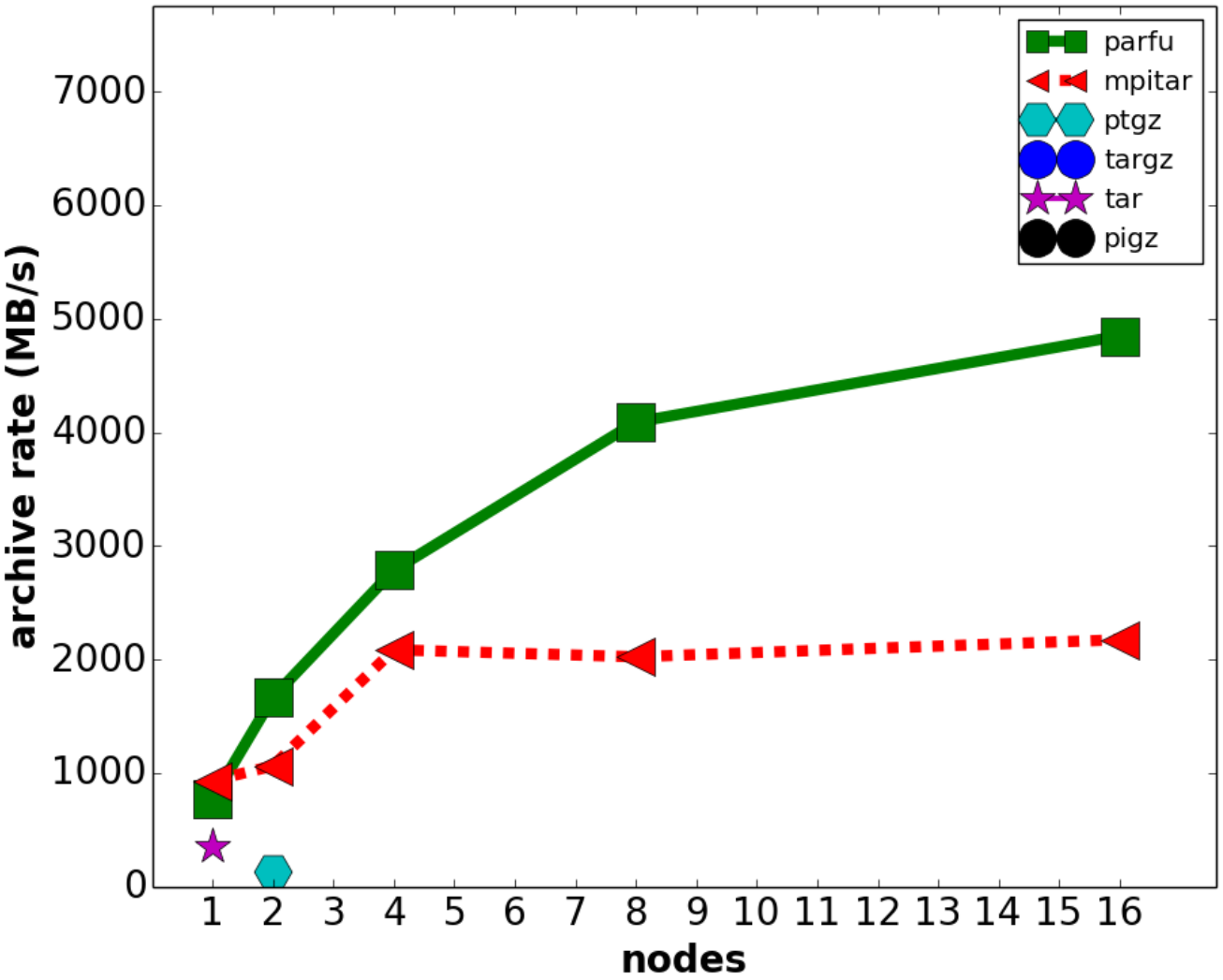
Performance: GWAS on Stampede 2



Performance: Arabidopsis on Stampede 2



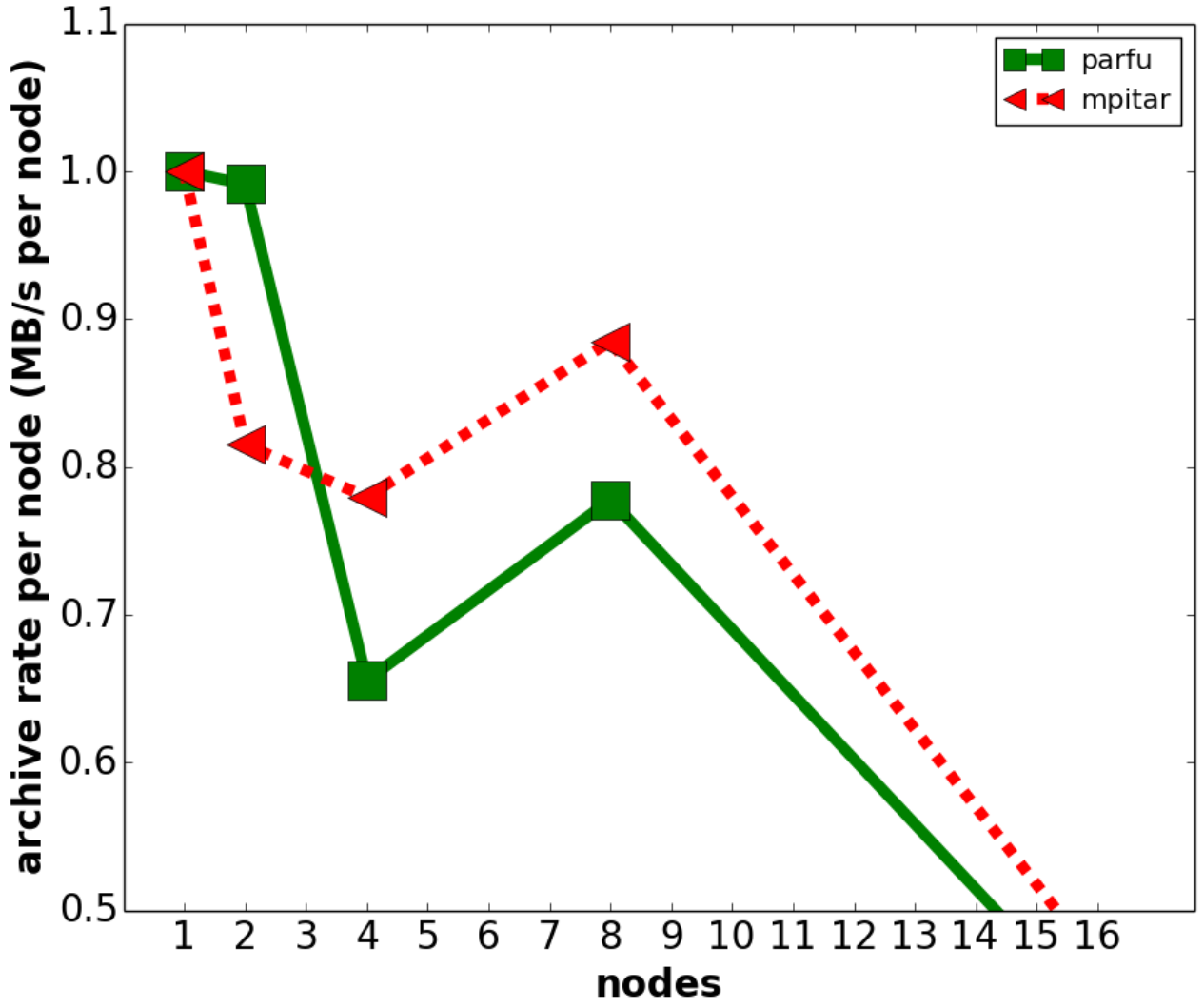
Performance: VarCall on Stampede 2



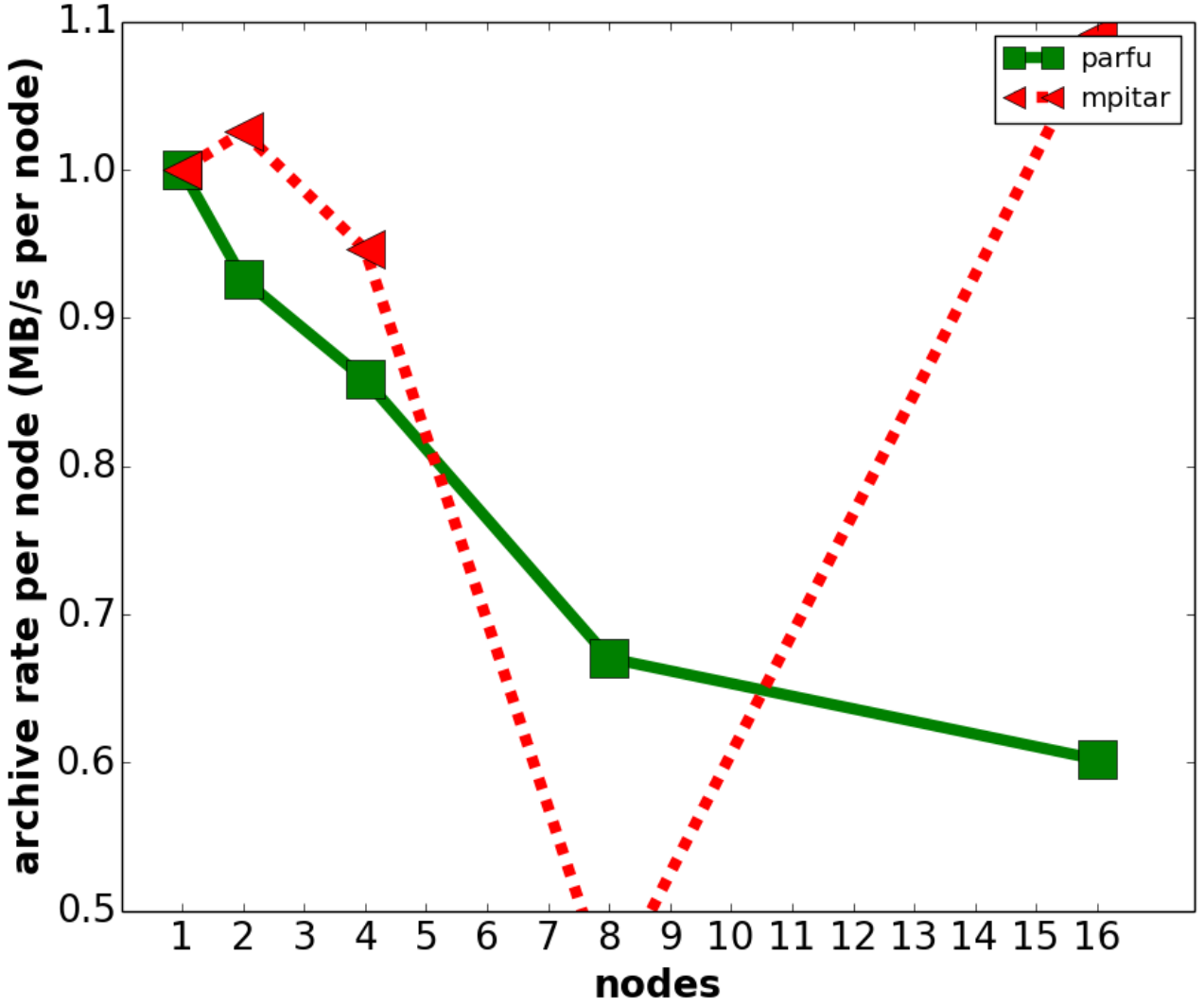
Scaling Graph Explanation

- Scaling graphs display performance as a function of nodes relative to the performance of the next lower node count
- Scaling value of 1 indicates perfect scaling: double the nodes means double the performance
- Scaling value of 0.5 indicates infinitely bad scaling: double the nodes doesn't improve performance **at all**
- Scaling value of ~0.7 or higher is "reasonable" scaling; it's mostly worth it to scale up this far

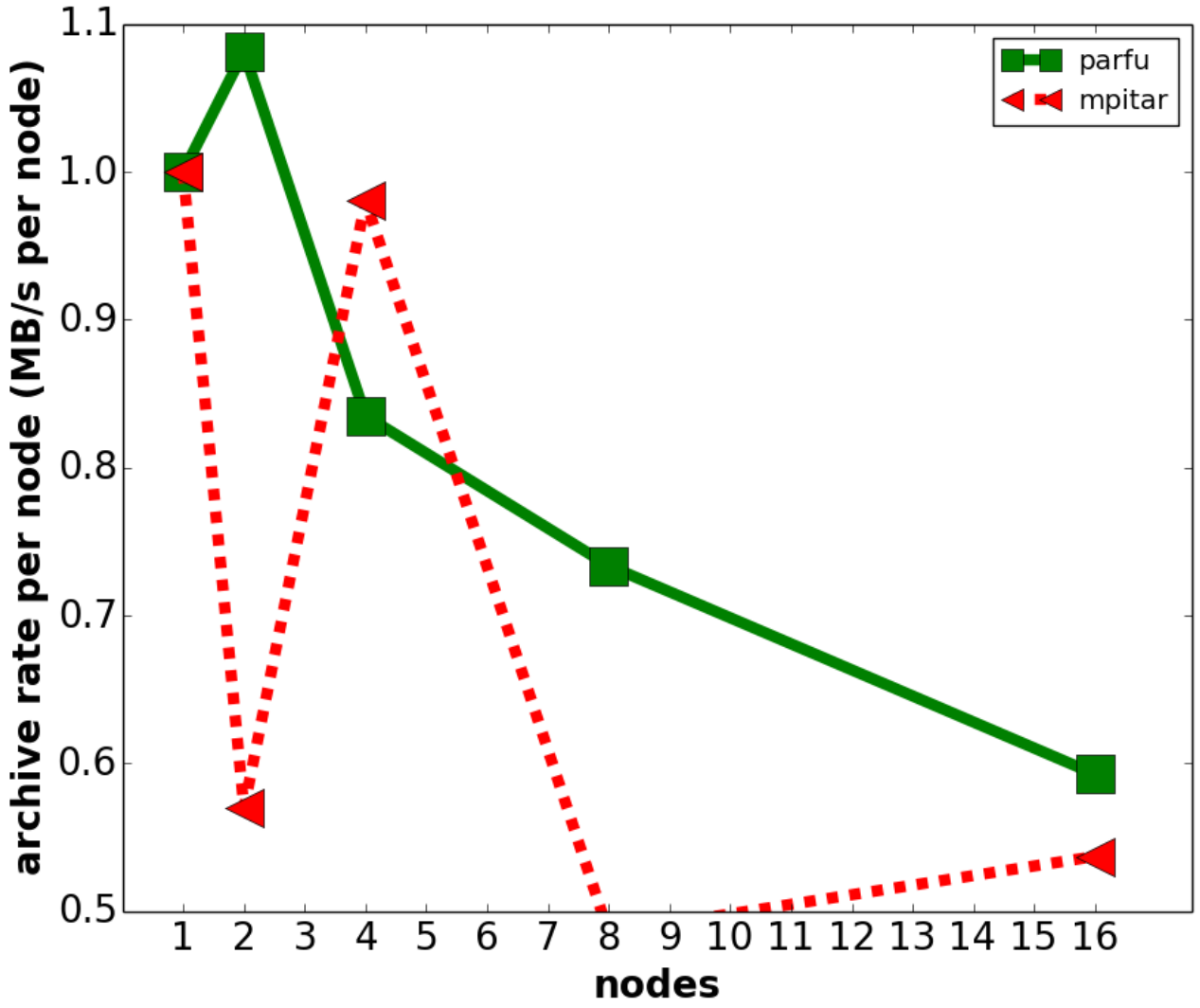
GWAS Scaling on Stampede 2



Arabidopsis Scaling on Stampede 2

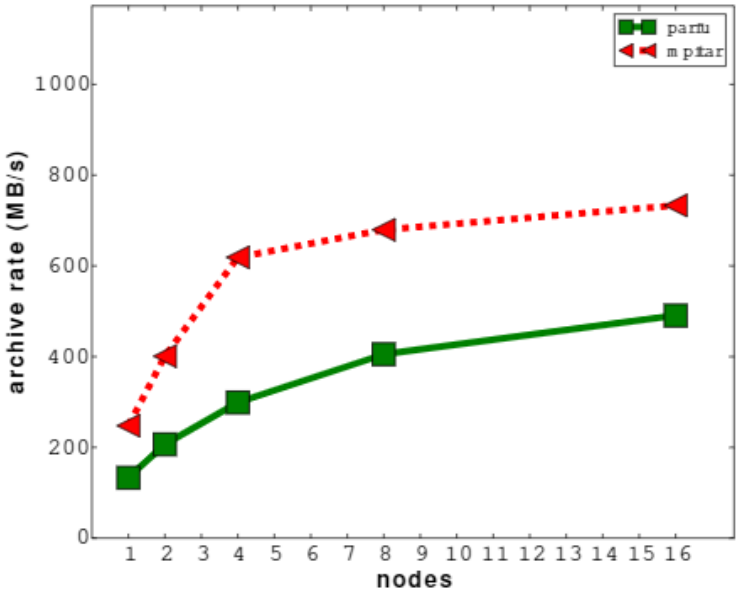


VarCall Scaling on Stampede 2

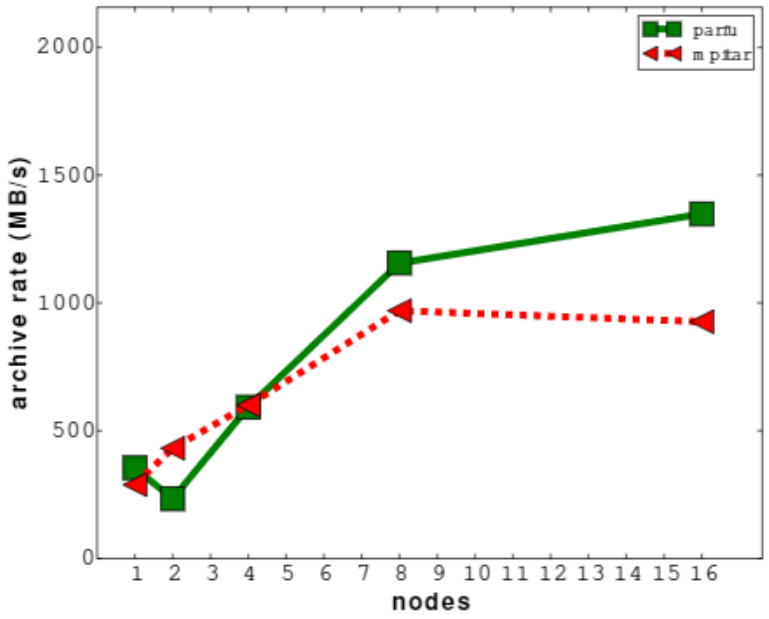


Performance on Blue Waters

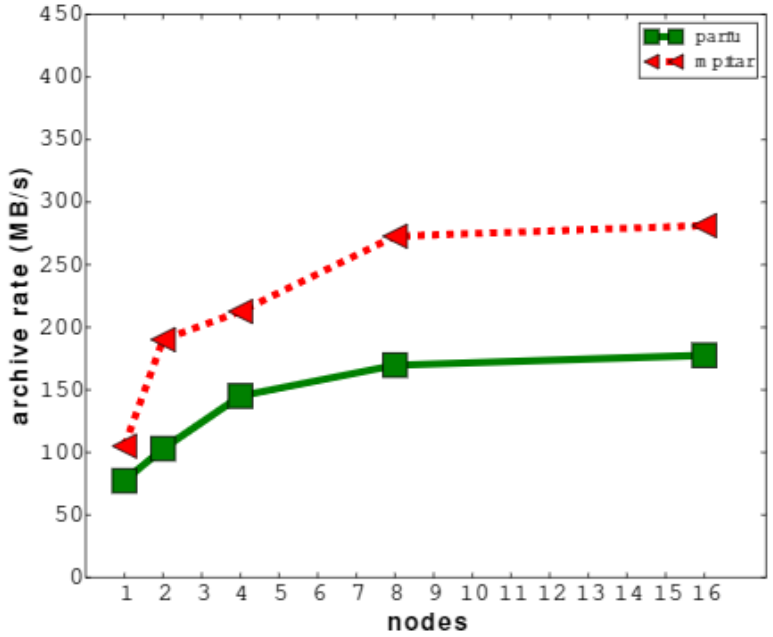
GWAS



ARABIDOPSIS



VARCALL



Overall Performance Conclusions

- Mpitar and parfu scale up to 4 nodes, sometimes 8
- They improve archiving performance ~10x faster than conventional tools
- Worth pursuing for future deployment and development

Success Story

"PK Yeung and the NFS PRAC allocation for *'A Penultimate Petascale Computational Laboratory for Turbulence and Magnetohydrodynamic Turbulence*' thank the Blue Waters collaboration for their assistance in moving our simulation results off the Blue Waters system at the end of our experimental runs. Our simulation snapshots consist of directories of files 4TB each. The parfu aggregation tool developed by the Blue Waters support team allowed us to package up our files so they were easy to transport to other systems."

Tool Status

- Ptgz: Development stopped, author has left NCSA
- Mpitar: In use as internal tool
- Parfu: In active development, plan for version 1.0 in C++ with tar-like interface in spring 2022

Parfu Feature Roadmap

- Full application (spring 2022)
  - Includes file listing
  - Extraction (including partial)
  - Full CLI user interface
- Compression?

Code Locations

- https://github.com/ncsa/NCSA-Genomics_ptgz
- https://git.ncsa.illinois.edu/rhaas/mpitar/-/tree/master
- https://github.com/ncsa/parfu_archive_tool
  - Also includes testing framework and scripts

Thanks

- NSF Grants: OAC-2004879 and OAC-1550514
- NSF OCI-0725070 and ACI-1238993
- NCSA Industry program: access to iForge
- NSF XSEDE allocation TG-ASC170008
- TACC discretionary allocation “parfu_Frontera”
- Blue Waters Project Office