

Update on Damaris: How to Make CM1 Scale Linearly up to 10K Cores And What Comes Next

Matthieu Dorier¹, Gabriel Antoniu²
KerData Research Team

¹ENS Cachan - Brittany, IRISA

²INRIA Rennes – Bretagne Atlantique

Joint work with Franck Cappello, Marc Snir, Leigh Orf

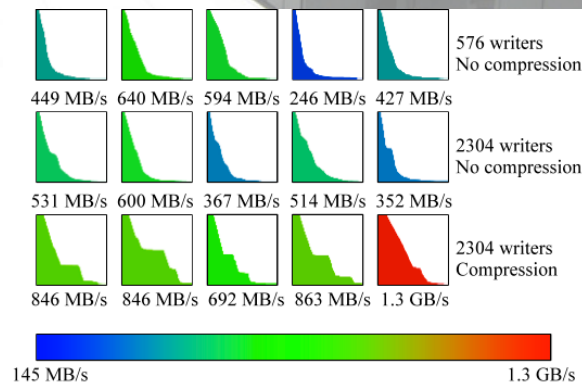
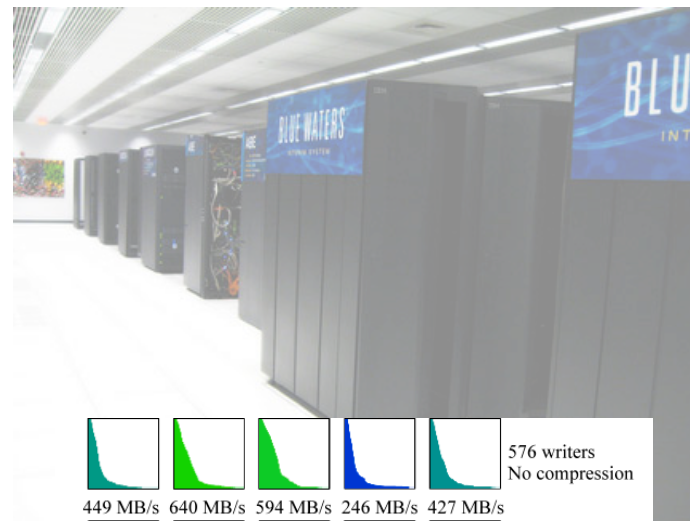
Sixth workshop of the
Joint INRIA/UIUC Laboratory for Petascale Computing
November 21-23, 2011

Outline

- Recall on the motivation
- Introduction to Damaris
- New results on Kraken with CM1
- Work in progress: visualization
- Potential directions using the BlobSeer approach
- Conclusion

Motivations

- Large-scale climate simulations on Blue Waters (e.g. Georges Bryan's Cloud Model 1)
- With great powers comes great responsibility
more data, always more data:
 - Terabytes of new data every minute
 - More I/O jitter, more unpredictable run time
 - More communication, higher pressure on I/O servers
 - More difficulties to read back, analyze, visualize...
- Need new I/O approaches allowing end-to-end scientific process on the same supercomputer, with negligible I/O and data processing overhead.

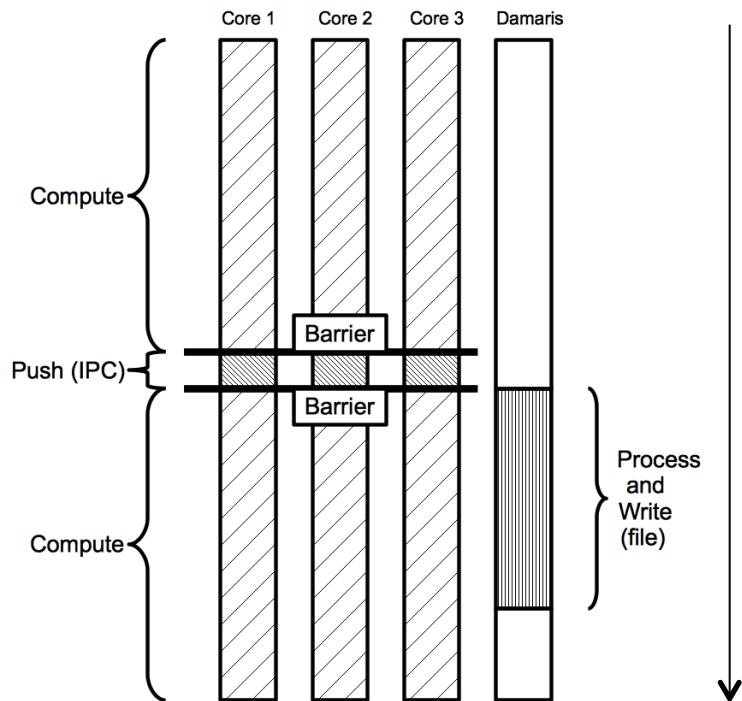


Outline

- Recall on the motivation
- **Introduction to Damaris**
- New results on Kraken with CM1
- Work in progress: visualization
- Potential directions using the BlobSeer approach
- Conclusion

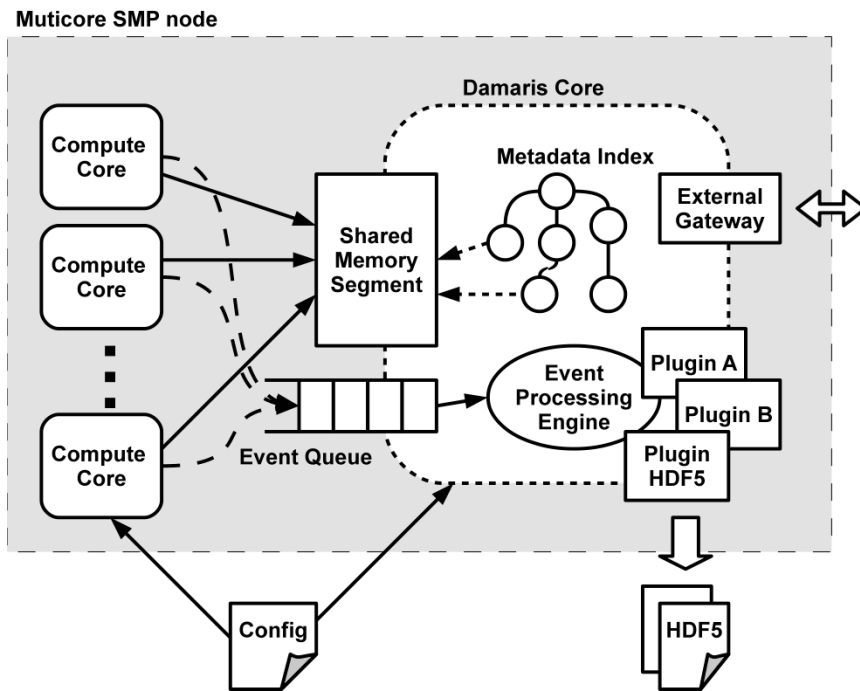
Damaris: Overview

- On multicore SMP nodes, using all the cores for computation may **not** lead to the best **performance**.
- Use **service-dedicated cores** on each SMP node to...
 - Perform I/O, data compression, filtering...
 - Data analytics and visualization



Damaris: Overview

- On **multicore SMP nodes**, using all the cores for computation may **not** lead to the best **performance**.
- Use **service-dedicated cores** on each SMP node to...
 - Perform I/O, data compression, filtering...
 - Data analytics and visualization



Design principles

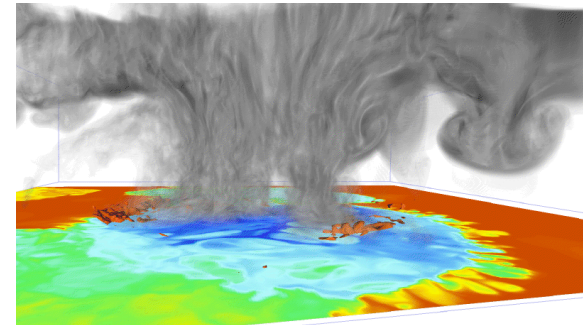
- Uses **shared-memory** between computation cores/dedicated cores
- **Avoids copy of data** as much as possible
- **Describes** the data externally
- Provides a **plugin system** to adapt to the user's needs

Damaris: Current State

- **Version 0.3.1** available at <http://damaris.gforge.inria.fr> along with **tutorials** and **documentation**;
- **Works on Linux** (including **Linux Cray**)
- Adapts to simulations written in **C**, **C++** and **Fortran**
 - Extremely simple API
 - Helper functions to integrate in **MPI applications** (communicators splitting, etc.)
- Uses **XML** files to describe the data and the plugins
 - Along with an XSD validation schema
- Plugins can be written in **C++** and in **Python**
 - Allows to use Python libraries such as **NumPy**, **SciPy**, **Matplotlib**, **YT**, ... to perform data analysis and visualization
 - Allows to change your plugins very easily by changing a **script file**, without recompiling, even without shutting down the application

The Kraken Cray XT5

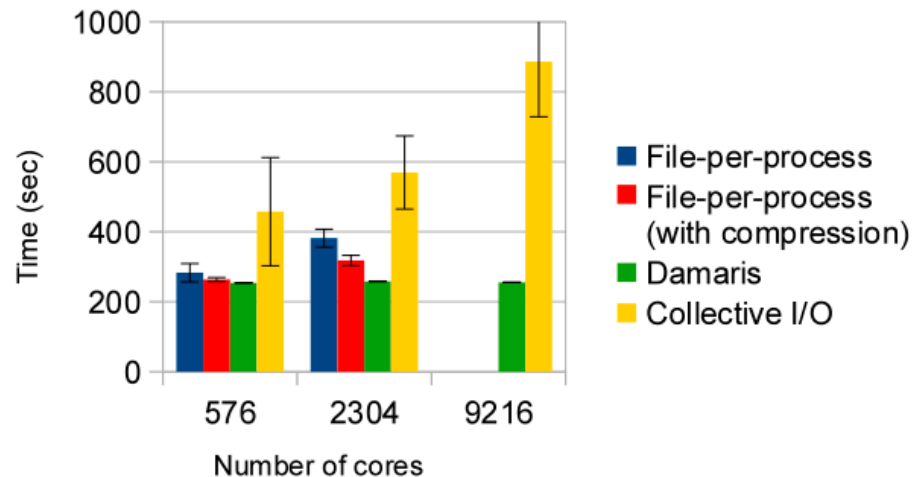
- Currently 11th in the Top500 (1,17 PetaFLOPs)
- 12 cores per node (112,896 cores), 16 GB local memory per node
- Lustre parallel files system: 48 OSS (336 OST), 1 MDS



- **Note:** just like Blue Waters, Kraken is provided by Cray, has a Lustre file system and works with Cray Linux Environment.
- *Results on this platform are highly relevant with regard to expected I/O behavior on Blue Waters.*

Results on Kraken with CM1

- **File-per-process:** creates too many files, becomes intractable starting from 2K cores
- **Collective I/O:** imposes huge I/O overhead (70% time spent in I/O), does not allow compression
- **Damaris**
 - 70% improvement with 9216 cores
 - No more I/O jitter
→ Predictable run time
 - 12 times fewer files, no synchronization
 - I/O scheduling
→ 15 times higher throughput
 - Overhead-free compression (600%)



CM1 weak scaling on Kraken

In-situ Visualization with Damaris

- **Goal: shorten the path from the running simulation to the retrieval of relevant scientific results, avoid moving data across different machines.**
- **Automate image generation using Python+Matplotlib, YT, SciPy**
 - Data published by the simulation to Damaris is visible to Python as a NumPy array
 - NumPy can transform the data, perform diagnosis, compute images using Matplotlib
 - By analyzing data, Damaris can select and draw what is interesting for the scientist
- **Interactive visualization using VisIt**
 - Visualization on demand: connecting VisIt to the running simulation WITHOUT perturbing it
 - Damaris acting as a rendering engine (e.g. use of NVIDIA's GPU on Blue Waters' compute nodes)
 - Capability to re-load old data to perform multi-simulations comparison

Outline

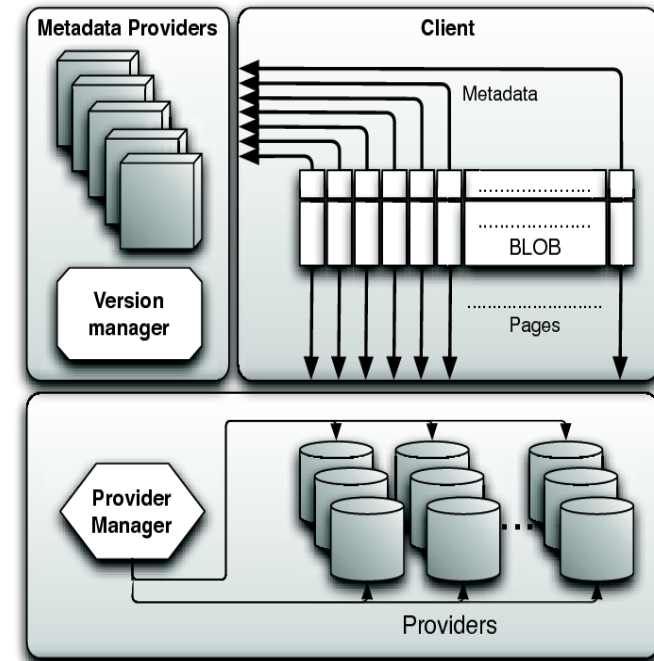
- Recall on the motivation
- Introduction to Damaris
- New results on Kraken with CM1
- Work in progress: visualization
- **Potential directions using the BlobSeer approach**
- Conclusion

Further Leverage the I/O Cores

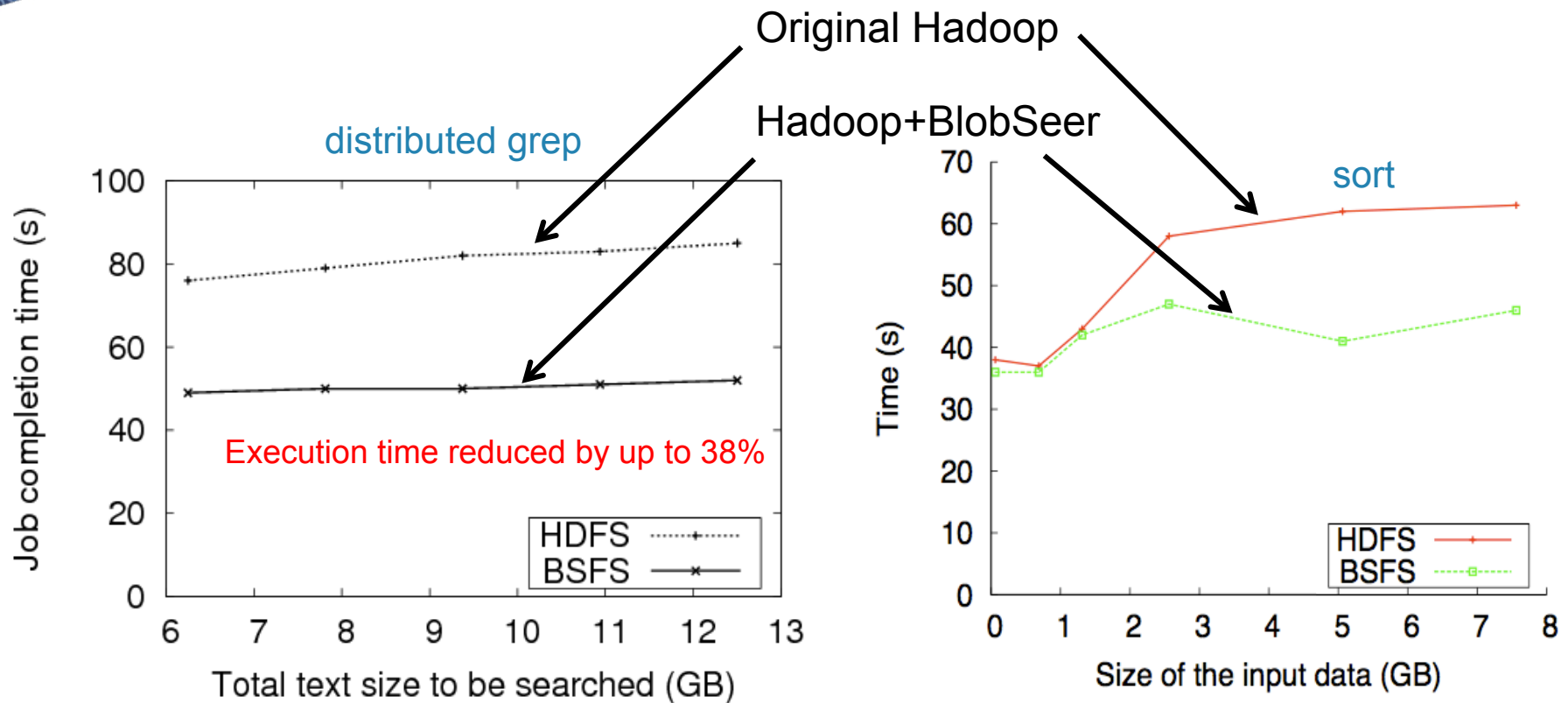
- Damaris can help to generate data digests (*metadata*) for specific visualization patterns
 - Some visualization requests could only rely on *metadata*
 - Impact: reduce concurrent accesses to shared *data*
 - Concurrent read/write accesses to *metadata*
 - Possibly involving historical metadata
- Need a metadata layer optimized for highly-concurrent accesses

BlobSeer: Concurrency-Optimized Data/Metadata Management

- BlobSeer: software platform for scalable, distributed BLOB management
- Developed by the KerData Team, INRIA, Rennes
 - Decentralized data storage
 - Decentralized metadata management
 - Versioning-based concurrency control
 - Lock-free concurrent writes (= insert new data/metadata)
- A back-end for higher-level data management systems
 - Short term: highly scalable distributed file systems
 - Middle term: storage for cloud services
- Methodology
 - Design and implementation of distributed algorithms
 - Experiments on the Grid'5000 grid/cloud testbeds
 - Validation with “real” apps on “real” platforms: Nimbus, Azure, OpenNebula clouds...



Highlight: BlobSeer Improves Hadoop



MapReduce: a natural application class for BlobSeer

• *Journal of Parallel and Distributed Computing (2011), IEEE IPDPS 2010*

The MapReduce ANR Project (2010-2014)

Goal: an optimized Map-Reduce platform for cloud infrastructures

Total cost: 3,1M€, ANR funding: 827K€

Partners

- INRIA - KerData team (Rennes) – leader
- INRIA - GRAAL team (Lyon), France
- Nimbus team, CI, Argonne National Lab/University of Chicago, USA
- University of Illinois at Urbana Champaign, USA
- Joint UIUC/INRIA Laboratory for Petascale Computing
- IBM Products and Solutions Center, Montpellier, France
- Institute of Biology and Chemistry of Proteins, Lyon, France
- MEDIT (SME), Palaiseau, France



Highlight: Efficient Checkpoint/Restart for HPC Applications on IaaS Clouds

Platform properties

- No parallel FS for persistency
- Lots (almost free) local storage

Application properties

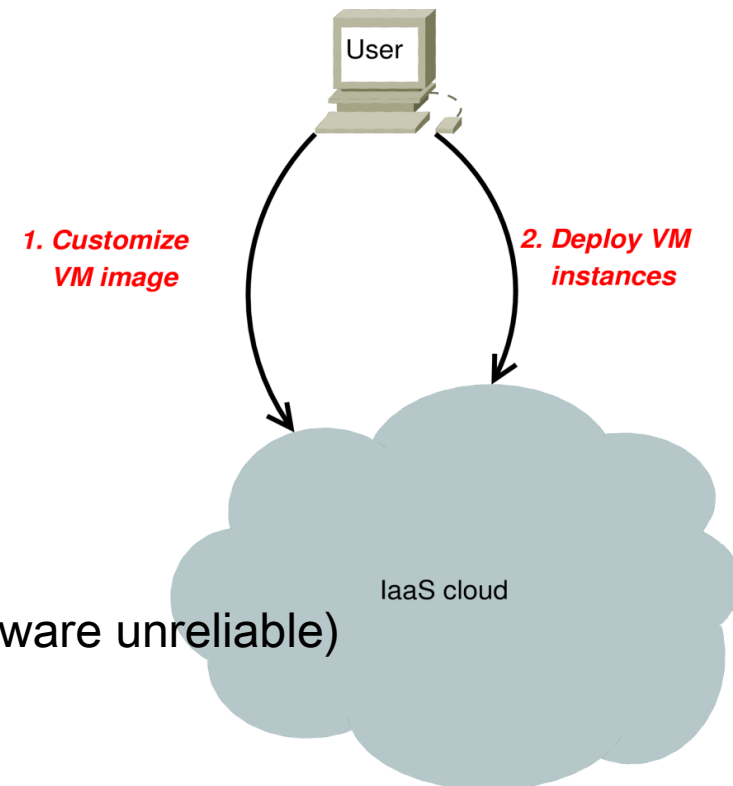
- Sharing of files is seldom needed
- Messages used for synchronization

Approach: rely on local storage

- Growing trend in HPC systems too

Challenges:

- Fault tolerance: low MTBF (commodity hardware unreliable)
- Suspend-Resume
- Migration



BlobSeer-based Checkpoint-Restart

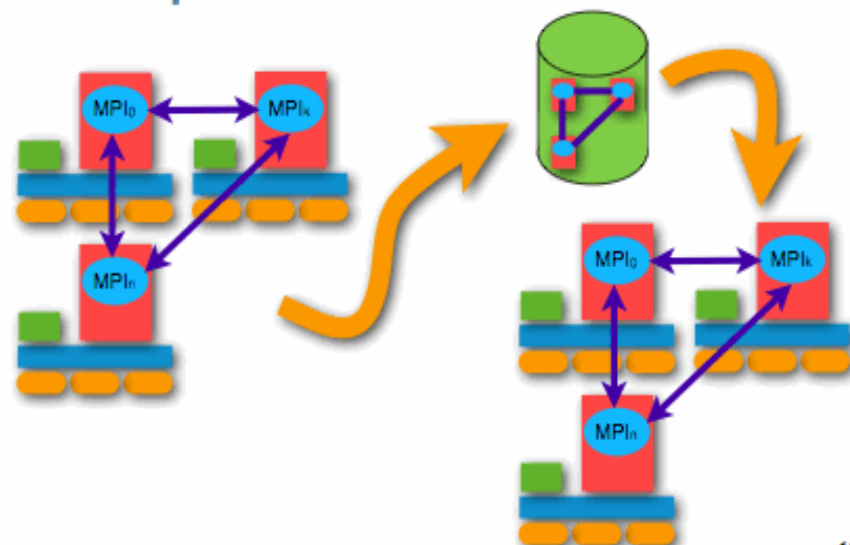
Why not *replication*?

- Costs are too high

Requirements

- Performance
- Little storage space
- Portability

Checkpoint Restart

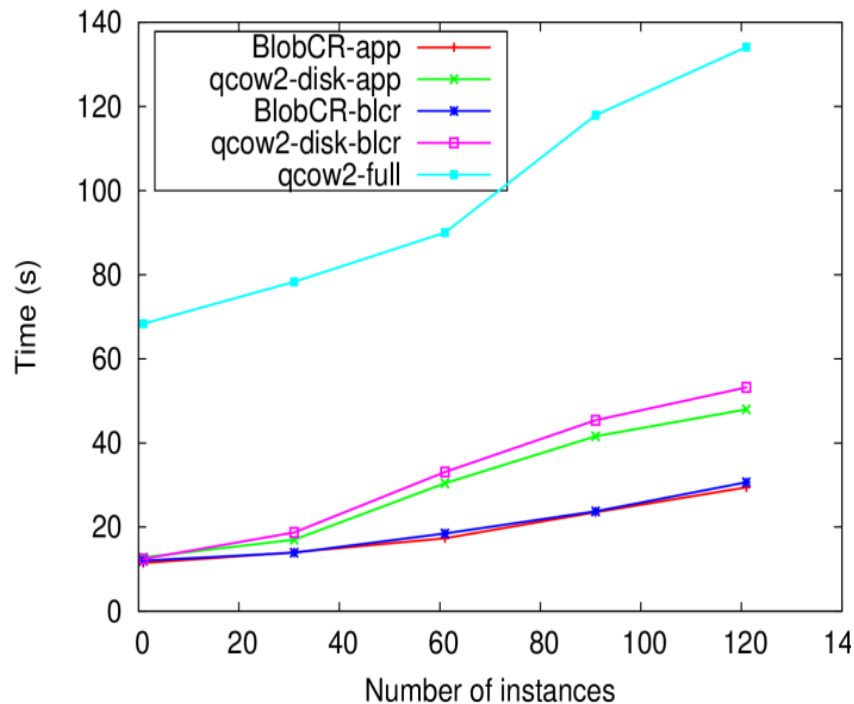


BlobCR - Virtual Disk Snapshots

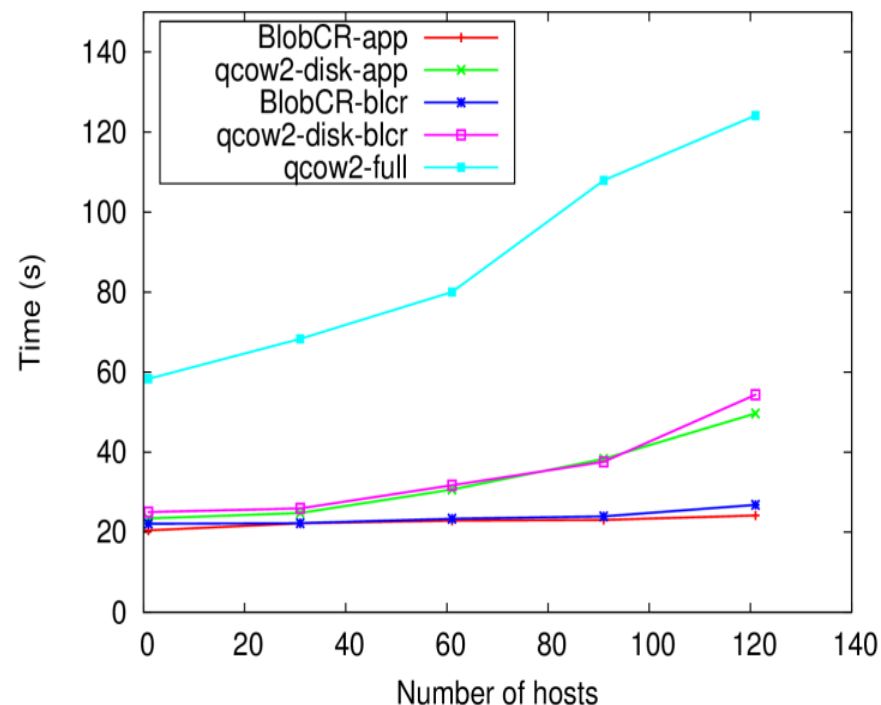
Idea: capture state of local FS

- Dedicated checkpoint repository ensures persistency
- “Magic” primitive for VM instances: **CHECKPOINT FS**
- Key advantage: roll-back FS changes
- Works at **process / application** level

C/R Performance



Speedup vs. PVFS + QCOW2:
8x VM level, **2x** process + app
level



Speedup vs. PVFS + QCOW2:
6x VM level, **3x** process + app
level

BlobSeer on Azure Clouds: A-Brain Collaboration with Microsoft Research

Application

- Large-scale Joint Genetic and Neuroimaging Data Analysis

Goal

- Assess and understand the variability between individuals

Approach

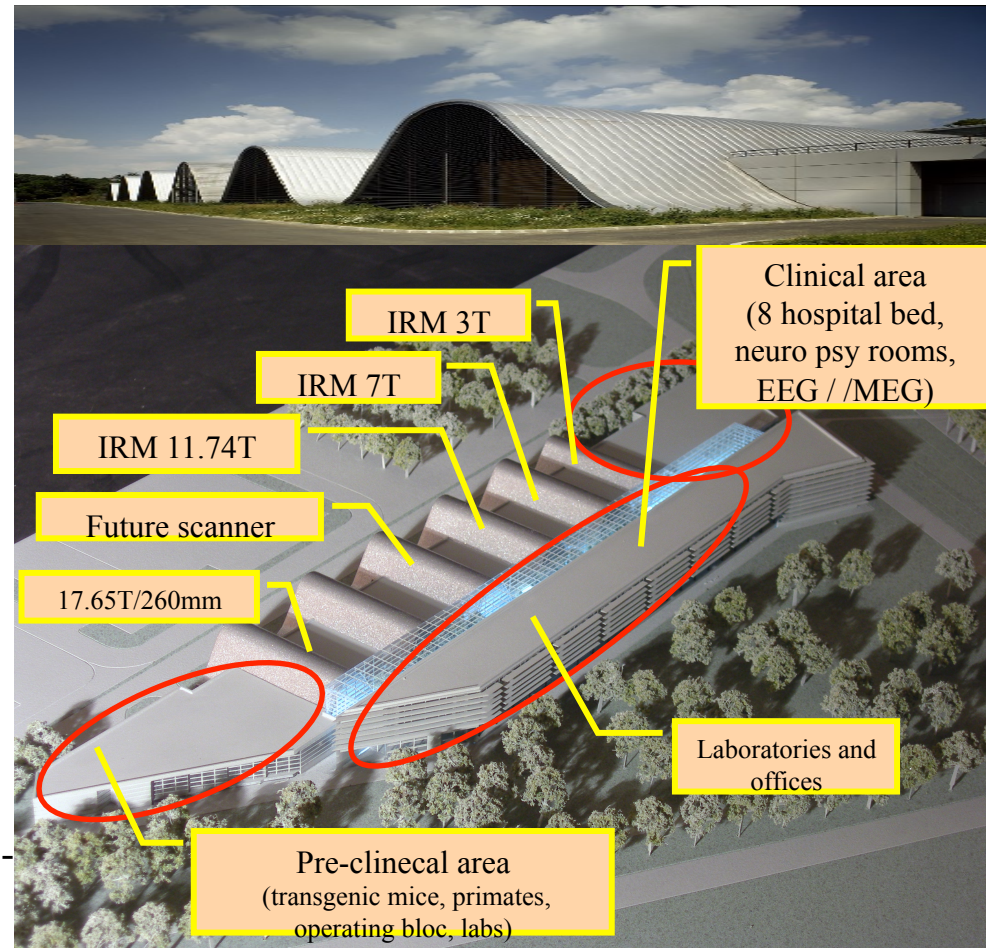
- Optimized data processing on Microsoft's Azure clouds based on INRIA's BlobSeer data management system

INRIA teams involved

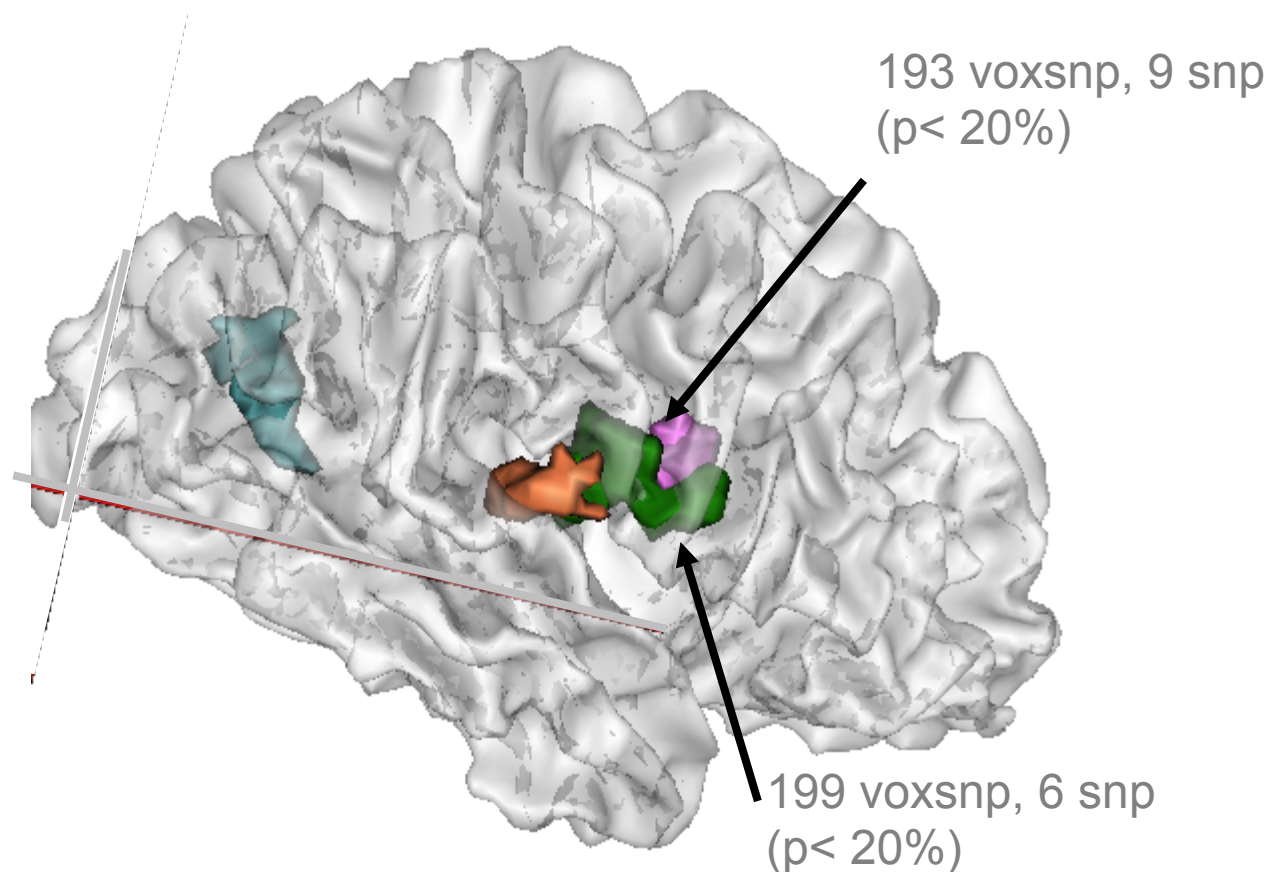
- KerData (Rennes)
- PARIETAL (Saclay)

Framework

- Joint MSR-INRIA Research Center
- MS involvement: Azure teams, EMIC



Illustration

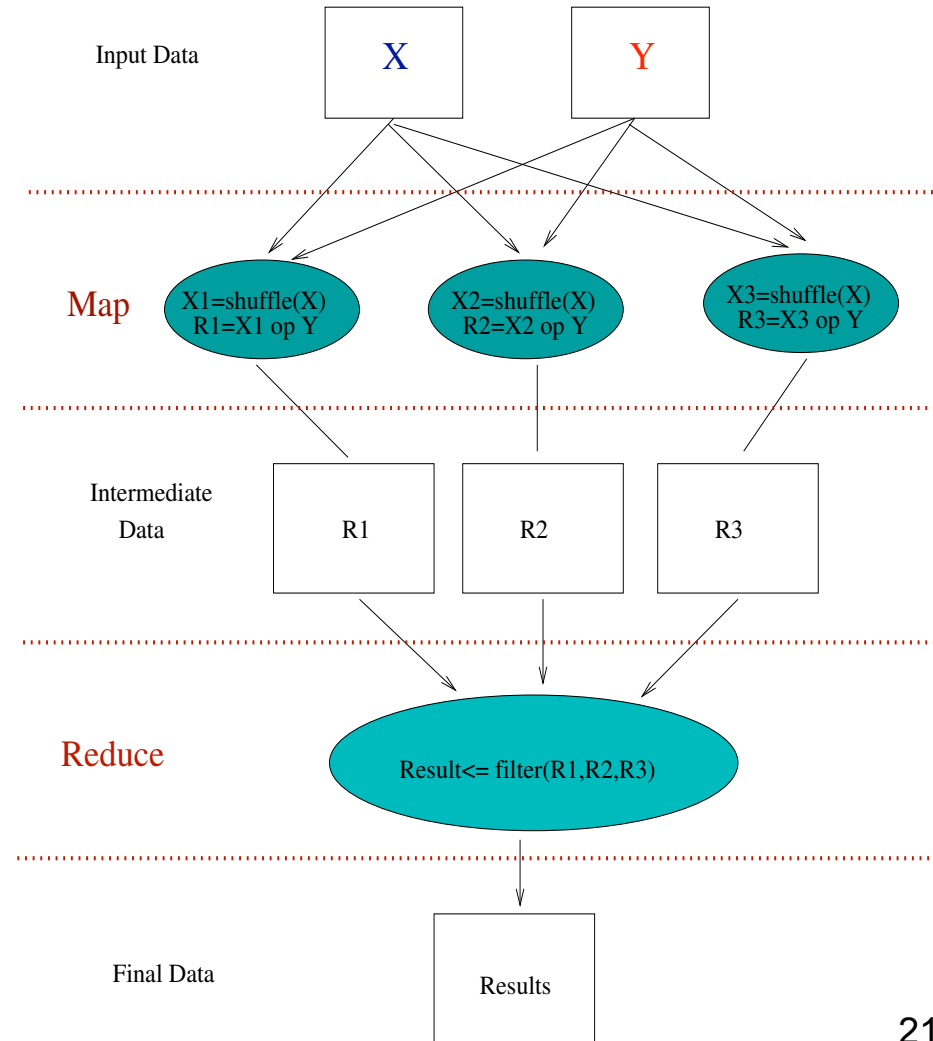


A-Brain: The goal is to reproduce this kind of study with 10^5 larger data

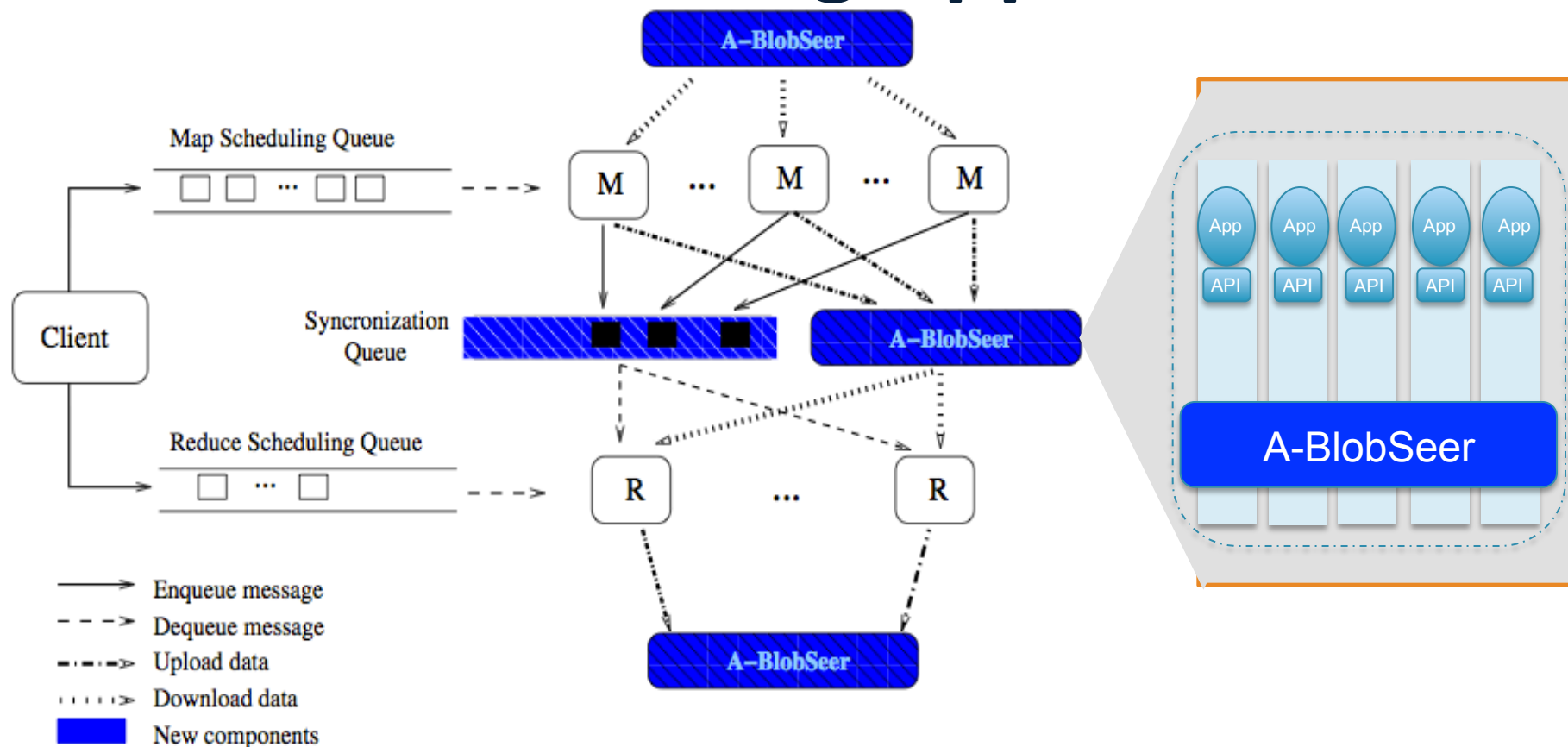
The Computational Problem

- Neuroimaging data (voxels in each contrast map): 10^5 to 10^6
- Genetic data: 10^6 variables
- Permutation tests: 10^3

Around 10^{15} tests

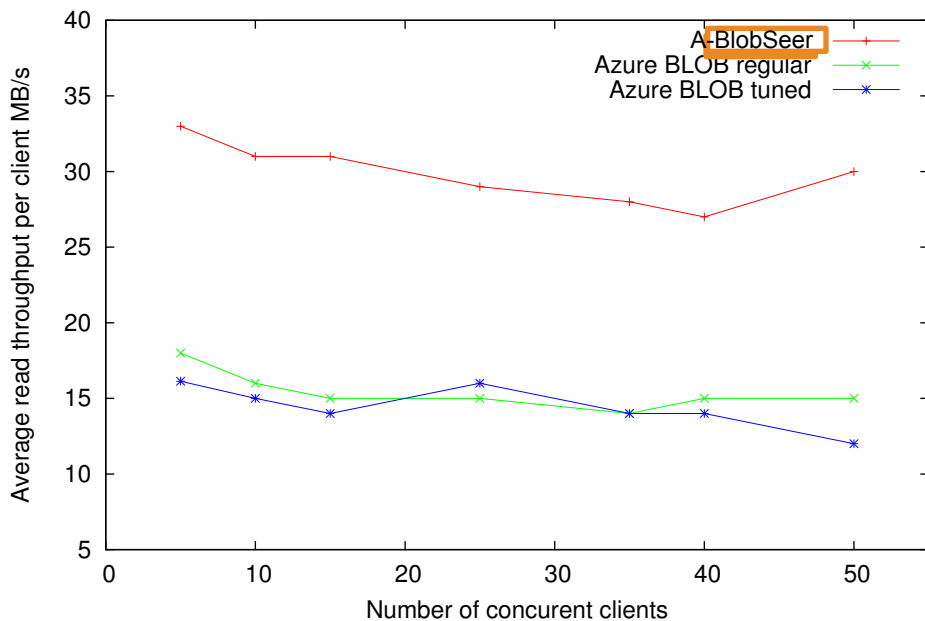


BlobSeer as a Storage Backend for Sharing Application Data



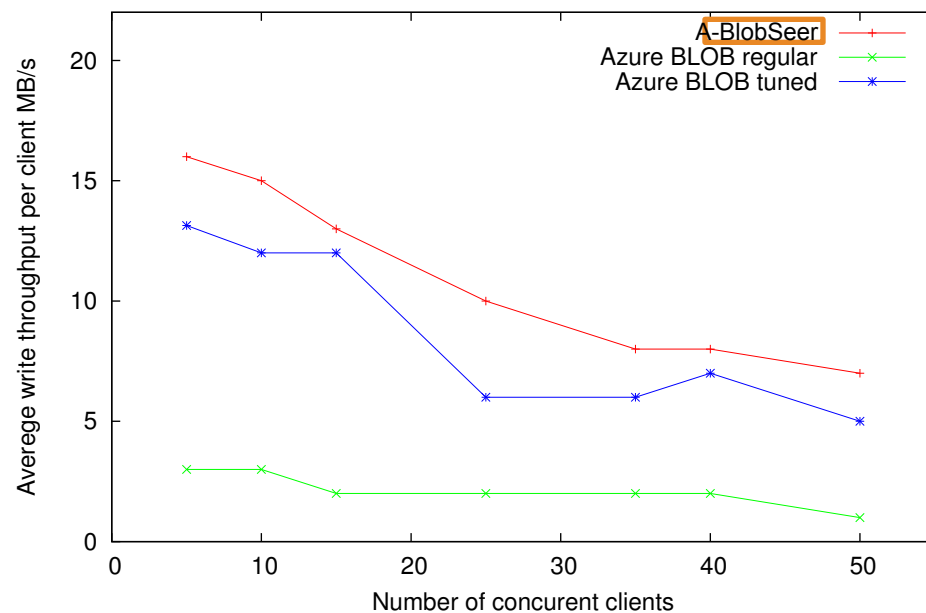
Throughput

Application pattern read throughput



read: **2.5x**

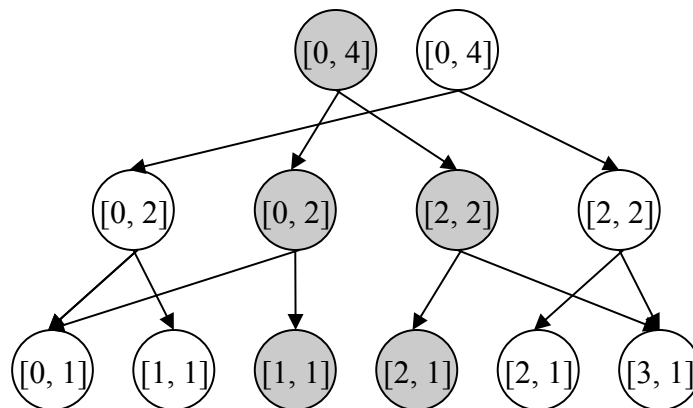
Application pattern write throughput



write: **3x**

Back to Blue Waters: Using BlobSeer with Damaris

- Leverage concurrency-optimized, versioning-based data management, apply it to *metadata* management for visualization
 - Metadata incrementally and dynamically updated by simulation
 - Multiple versions of the data stay available and can be used for advanced post-processing or visualization
 - Scalable concurrent reads and writes to metadata without sync



Next Steps: Concurrency-Optimized Metadata Management

- Leveraging BlobSeer: what is needed?
 - Adapt BlobSeer's metadata layer to the needs of visualization
 - Efficient file system back-end for BlobSeer
- What can we expect?
 - Enhanced support for inline visualization
 - Only from metadata, with no sync
 - From the I/O cores, with no sync

Concluding Remarks

- Damaris has proven to be very efficient on Kraken
 - 70% speedup on 9216 cores using the CM1 application
 - 15 times higher throughput, 600% overhead-free compression, I/O jitter hidden
 - CM1 now scales perfectly well, as opposed to using standard I/O approaches
 - Kraken is similar to Blue Waters' new Cray architecture
- Work in progress targets in-situ visualization
 - Using Python libraries to ease the design of visualization plugins
 - Using VisIt for interactive visualization
- Future work
 - Explore the potential of BlobSeer techniques for concurrency-optimized metadata
 - Simplifying end-to-end scientific process using a configurable workflow from scientific simulations to visualization and analysis tools

Concluding Remarks

- Damaris has proven to be very efficient on Kraken
 - 70% speedup on 9216 cores using the CM1 application
 - 15 times higher throughput, 600% overhead-free compression, I/O jitter hidden
 - CM1 now scales perfectly well, as opposed to using standard I/O approaches
 - Kraken is similar to Blue Waters' new Cray architecture
- Work in progress targets in-situ visualization
 - Using Python libraries to ease the design of visualization plugins
 - Using VisIt for interactive visualization
- Future work
 - Explore the potential of BlobSeer techniques for concurrency-optimized metadata
 - Simplifying end-to-end scientific process using a configurable workflow from scientific simulations to visualization and analysis tools

Thank you, questions?