Toward new I/O Approaches for Scalable Post-petascale HPC Simulations

Matthieu Dorier
ENS Cachan Brittany extension
matthieu.dorier@eleves.bretagne.ens-cachan.fr
KerData Team
Advised by **Gabriel Antoniu** and **Luc Bougé**

INRIA NCSA ENS CACHAN BRETAGNE UMR IRISA

Let’s start with pictures

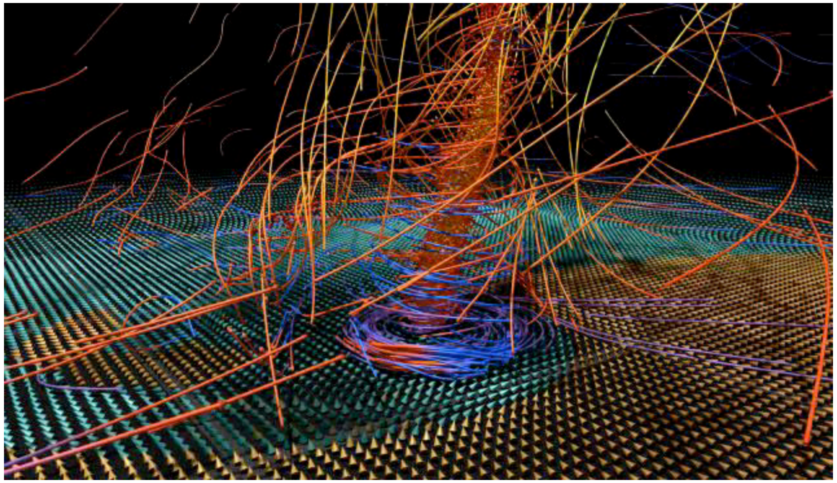
April 14 to 16, 2011 – Tornado outbreak in USA – 43 deaths

Let’s start with pictures

April 14 to 16, 2011 – Tornado outbreak in USA – 43 deaths

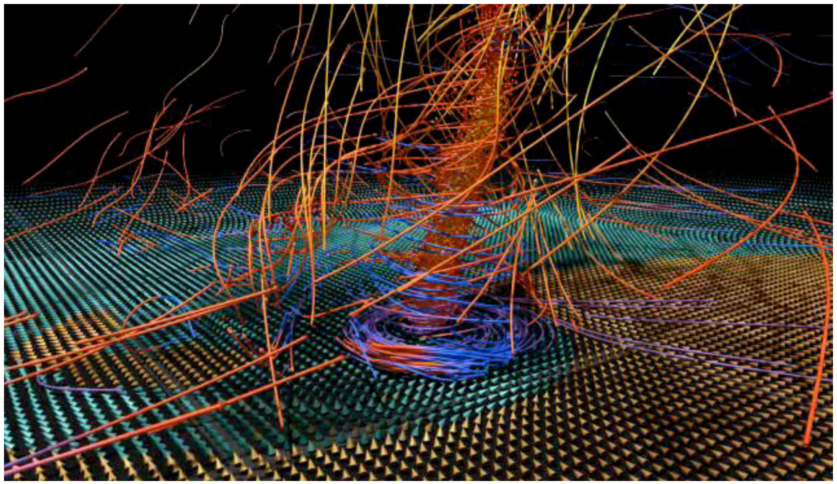
Could we have predicted this?

Understanding climate



- ✧ Large-scale simulations help understanding climate
- ✧ Require high performance

HPC simulations on BlueWaters



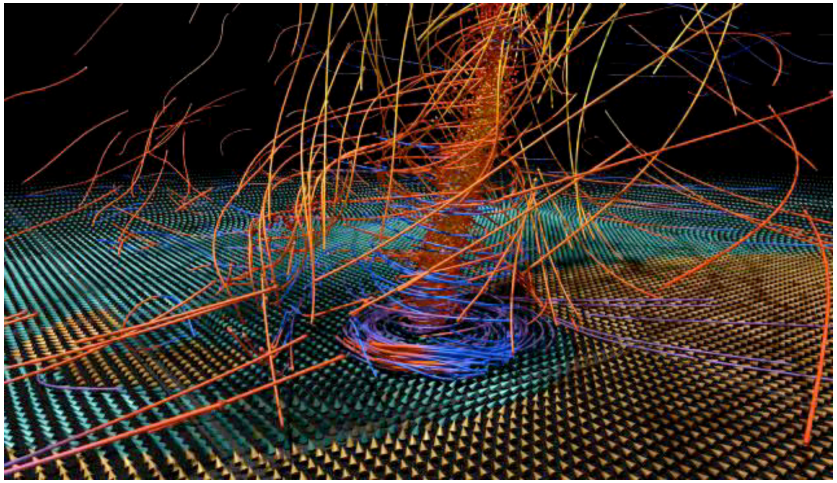
Context: *Joint Laboratory for Petascale Computing*, targeting **Blue Waters**

✧ More than 300.000 cores

✧ 11 petaflops (10^{15} op/sec) peak performance

(http://www.ncsa.illinois.edu/BlueWaters/)

HPC simulations on BlueWaters



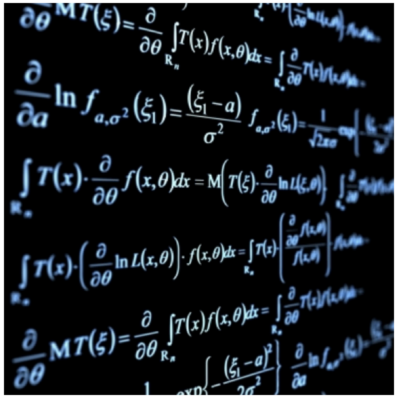
Context: *Joint Laboratory for Petascale Computing*, targeting **Blue Waters**

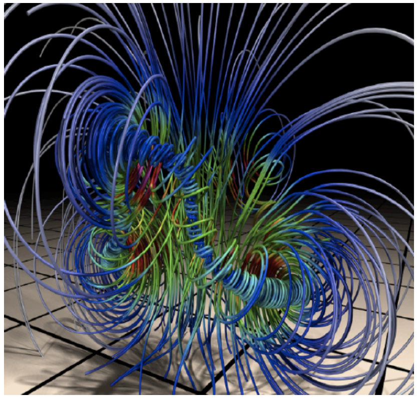
✧ More than 300.000 cores

✧ 11 petaflops peak performance

✧ **Simulations generating extremely large amounts of data** (terabytes every minute)

How to handle such large amounts of data?





- ✧ How to efficiently **store** and **move** data?
- ✧ How to **index**, **process**, **compress** these data?
- ✧ How to **analyze**, **visualize** and **understand** them?

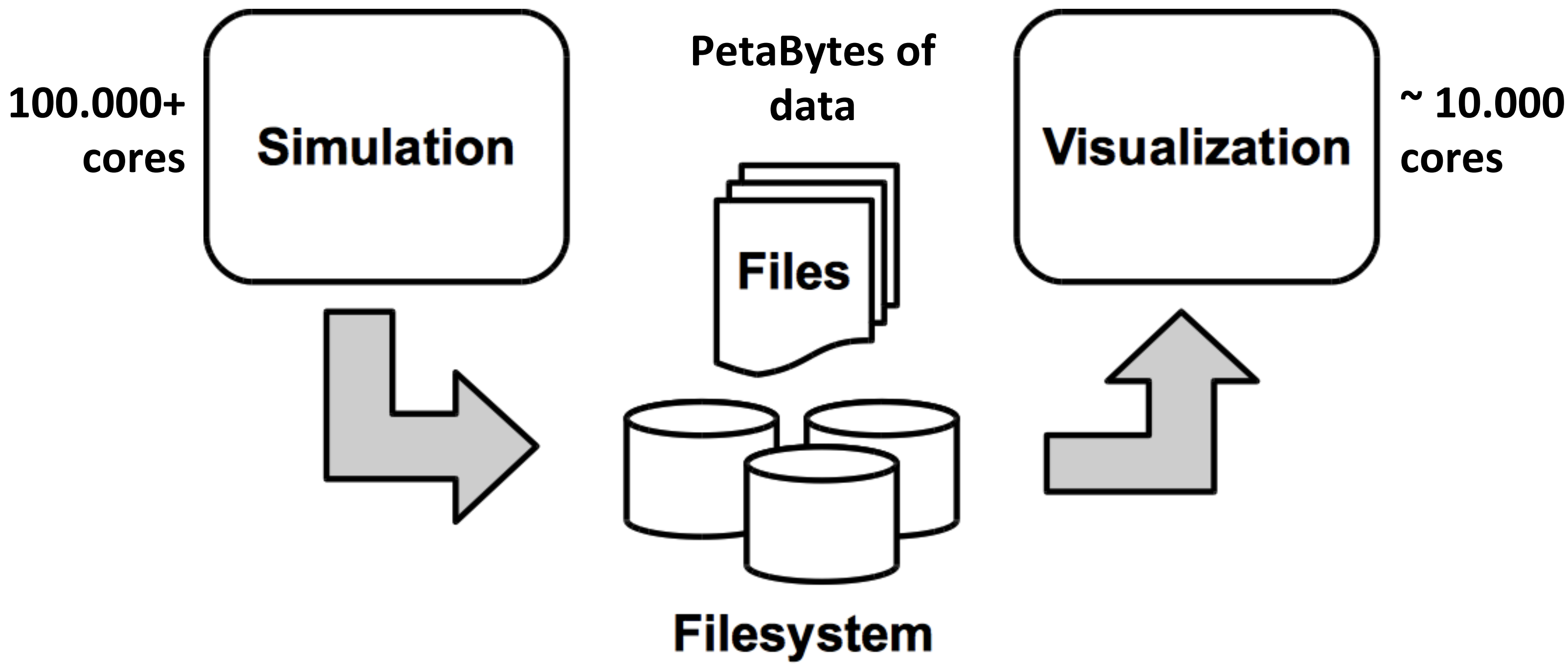
Outline

1. I/O and data management in HPC
2. Understanding I/O jitter
3. Damaris: our new approach to I/O
4. Experimental evaluations
5. Conclusion

Outline

1. I/O and data management in HPC
2. Understanding I/O jitter
3. Damaris: our new approach to I/O
4. Experimental evaluations
5. Conclusion

Standard I/O flow



- ✧**Periodic** data generation from the simulation
- ✧Storage in a **parallel file system**
- ✧**Offline** analysis and visualization [Childs,2010]

The key component: Parallel File Systems

[Carns et al.,2000]

[Schmuck et al.,2002]

[Donovan et al.,2003]

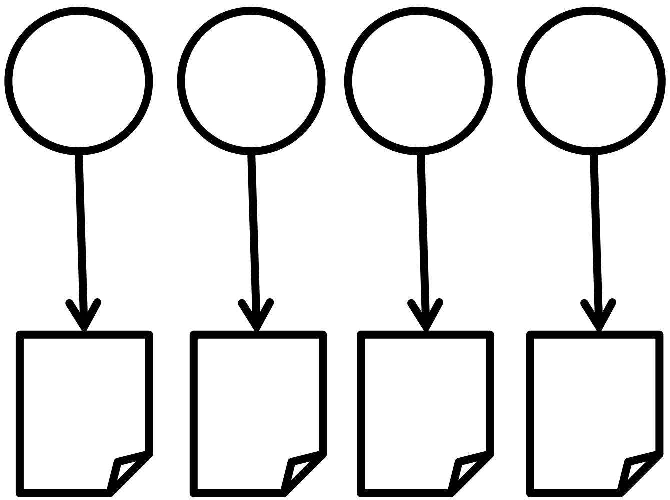
✧ Deployed on a set of **dedicated servers**

✧ **Shared** by all users (e.g. 100 GPFS servers on Blue Waters)

✧ Breaks files in **chunks** distributed across servers

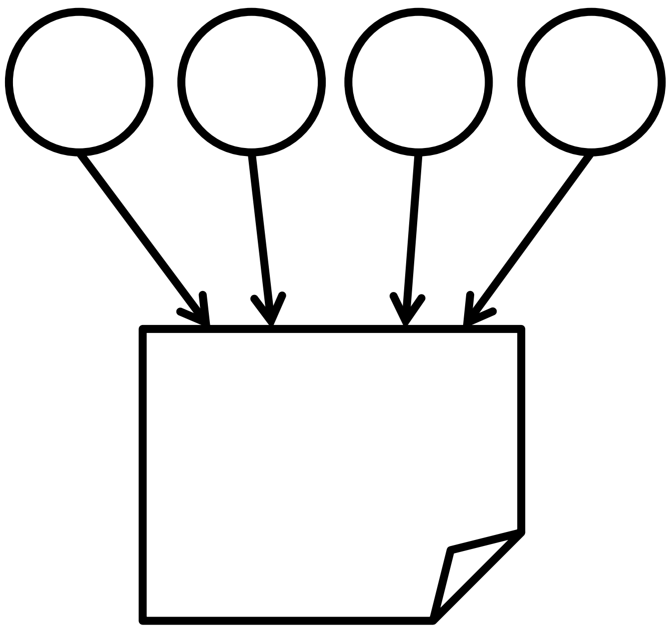
Handling I/O in simulations: Two main approaches

Independent I/O (file-per-process)



✧ Huge metadata overhead
✧ Hard to read back
✧ Easy (natural) to implement

Collective I/O



✧ Requires synchronization
✧ Hard to implement
✧ Optimizes communications

Problem #1: Unbalanced load, Periodic bursts of I/O

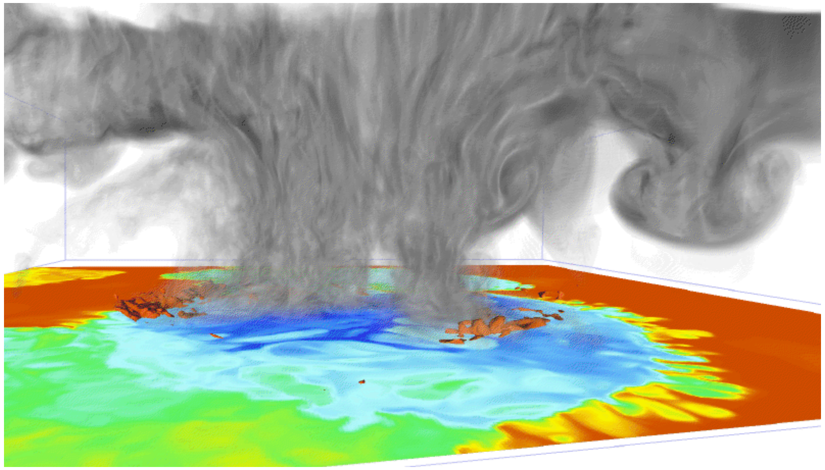
"Cardiogram" of a data server
(network activity when running a simulation)

Problem #2: I/O bottleneck

- **Too many files**: pressure on the metadata servers
  (e.g. Blue Waters 300.000 files/min)
- **Too much data**: pressure on the data servers
  (e.g. several Terabytes per minute)

Problem #3: data analysis

- ✧ All data are not useful
- ✧ How to process data, adapt data layout?
- ✧ When, where and how to perform visualization?



- ✧ **From offline visualization to inline visualization?**

Problem #4: I/O jitter

Outline

1. I/O and data management in HPC
2. Understanding I/O jitter
3. Damaris: our new approach to I/O
4. Experimental evaluations
5. Conclusion

I/O variability (or “jitter”)

Variability = difference between write time

✧From a process to another

✧From a write phase to another

Leads to unpredictable run time!

I/O variability (or "jitter")

Variability = difference between write time

✧From a process to another

✧From a write phase to another

Leads to unpredictable run time!

Origins of jitter

✧Network and file system contentions between processes

  ✧Internal interferences (processes of the same application)

  ✧External interferences (cross-applications)

I/O variability (or “jitter”)

Variability = difference between write time

✧From a process to another

✧From a write phase to another

Leads to unpredictable run time!

Origins of jitter

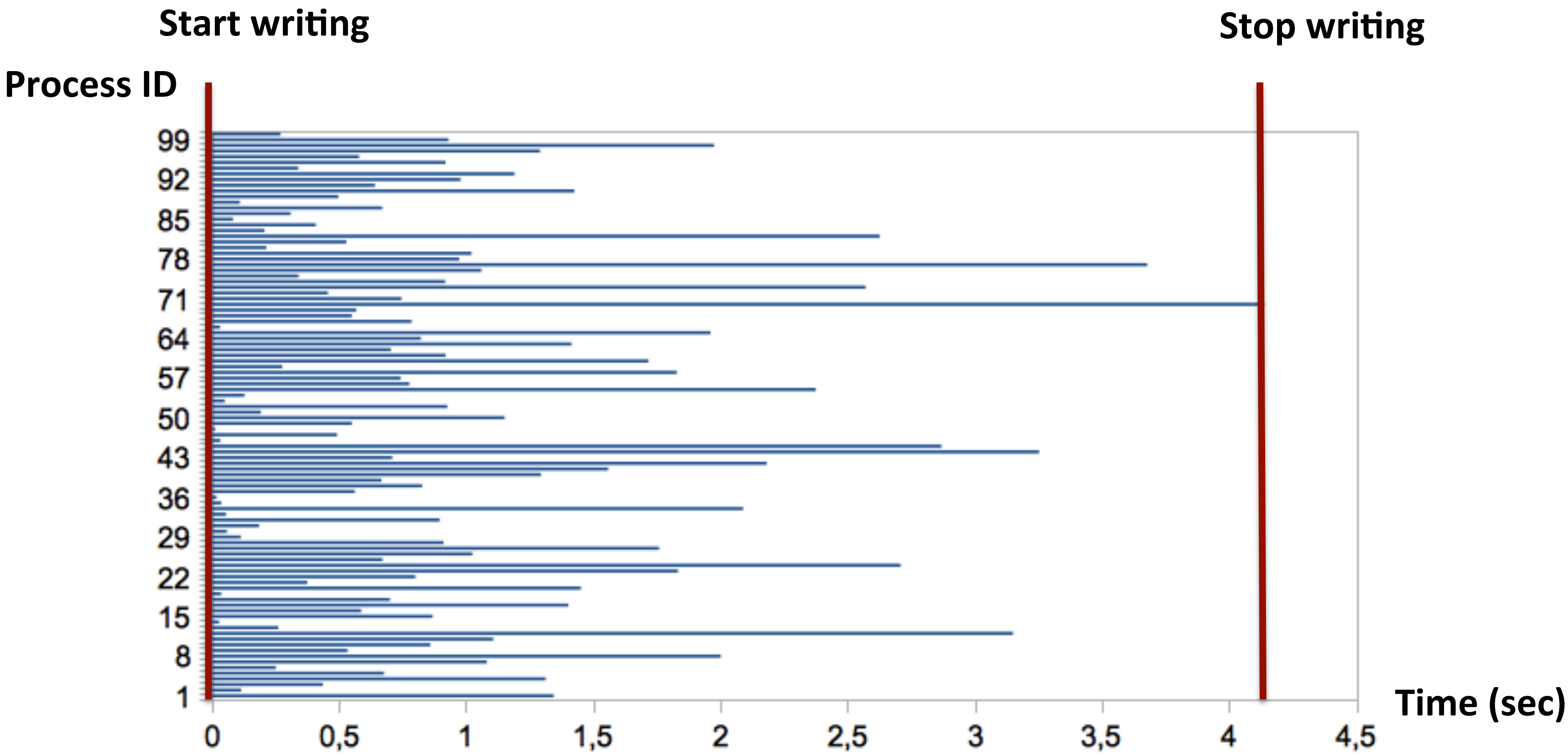
✧Network and file system contentions between processes

- ✧Internal interferences (processes of the same application)
- ✧External interferences (cross-applications)

Understanding the jitter

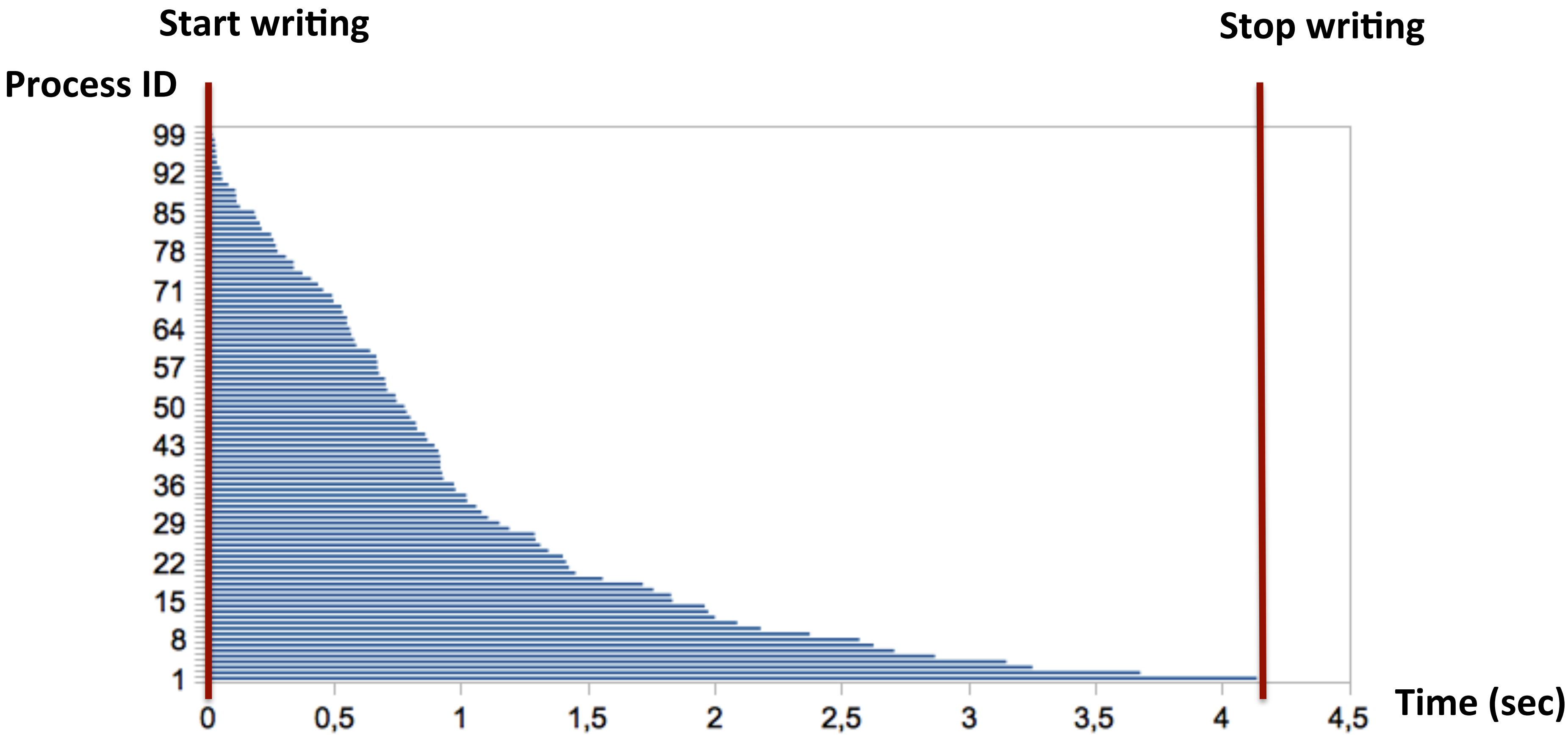
✧Statistical analysis (variance): intractable

How to interpret I/O variability?



Average = *X*, Standard deviation = *Y* … OK, and?

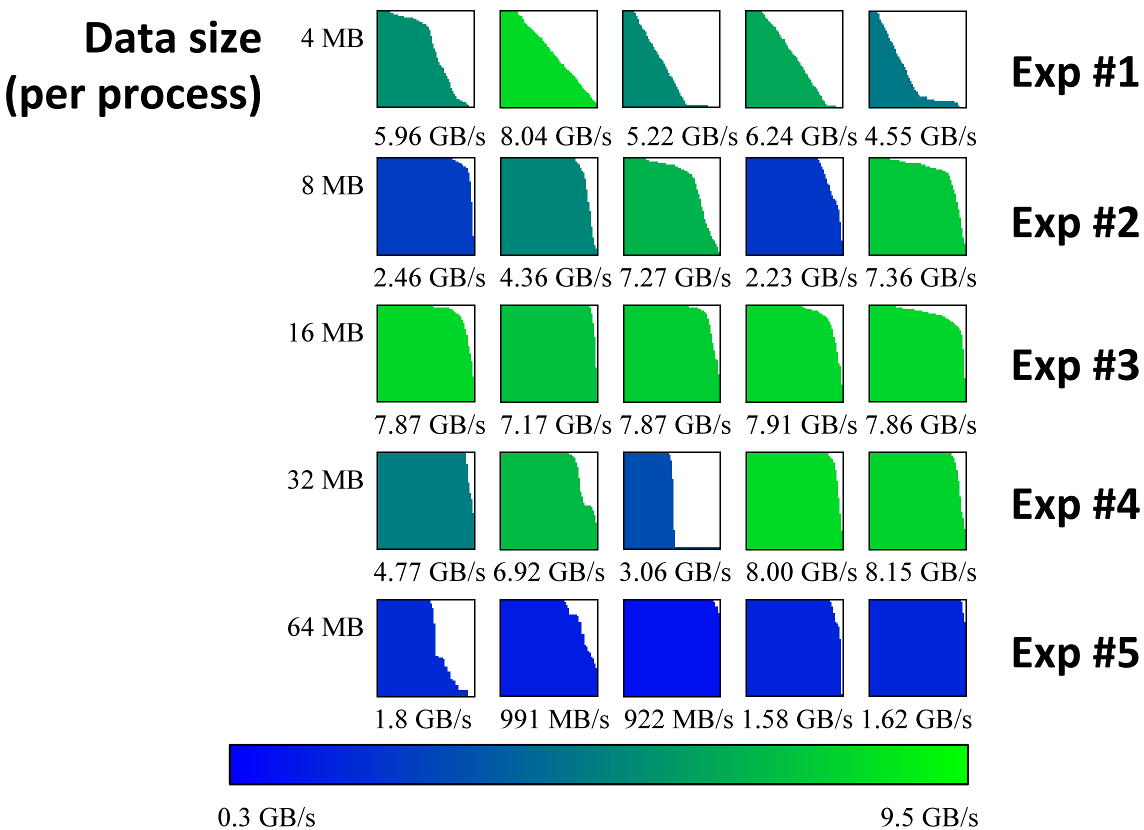
How to interpret I/O variability?



Let's sort this trace

- *White part corresponds to wasted time*

Contribution #1: Graphical comparison of traces



Color scale for aggregate throughput

Contribution #1: Summary

✧ A methodology to **visualize** and **interpret** variability

✧ Used on a set of 400 experiments (230400 mesures)

✧ **Validity**: from 400 MB/s to 21 GB/s on Grid'5000
(with PVFS and the IOR benchmark) by playing with 3 different parameters

Contribution #1: Summary

✧ A methodology to **visualize** and **interpret** variability

✧ Used on a set of 400 experiments (230400 mesures)

✧ **Validity**: from 400 MB/s to 21 GB/s on Grid'5000 (with PVFS and the IOR benchmark) by playing with 3 different parameters

*Tuning a parallel file system is hard, couldn't we simply **hide this variability**?*

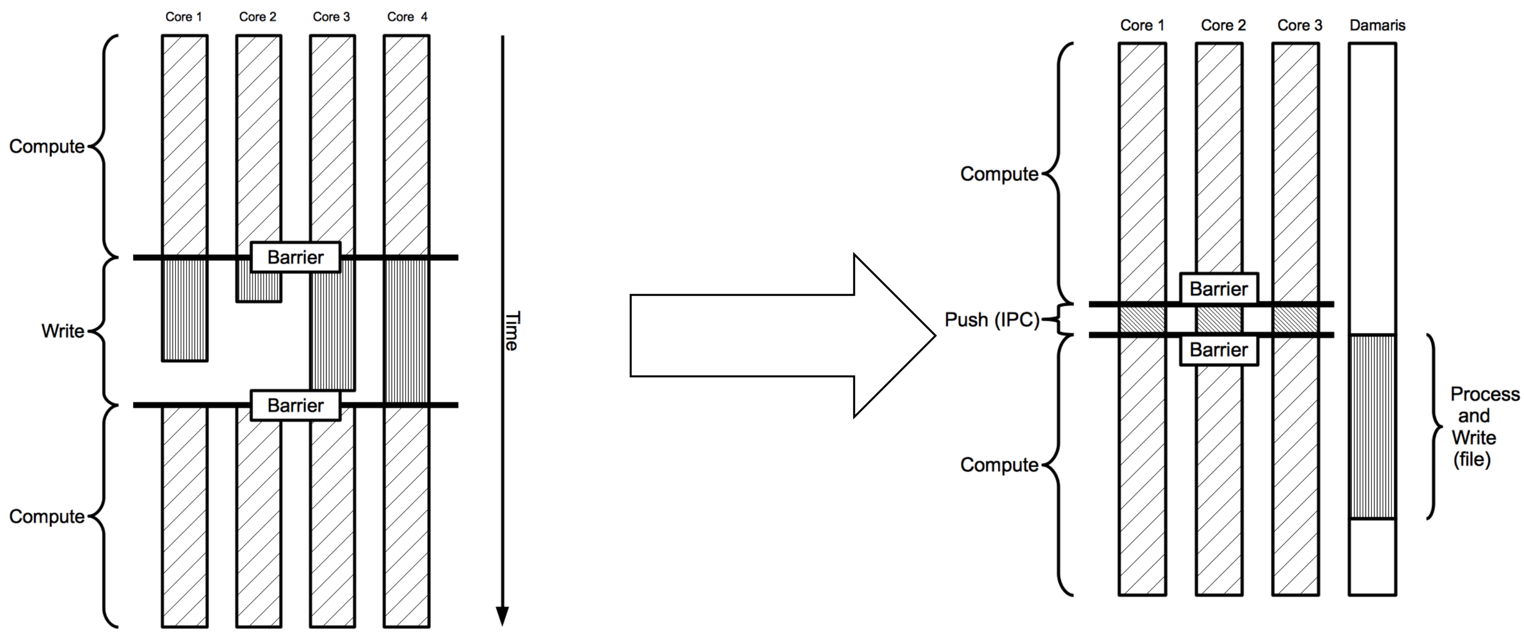
Outline

1. I/O and data management in HPC
2. Understanding I/O jitter
3. Damaris: our new approach to I/O
4. Experimental evaluations
5. Conclusion

On multicore SMP nodes…

Leave a core, go faster!

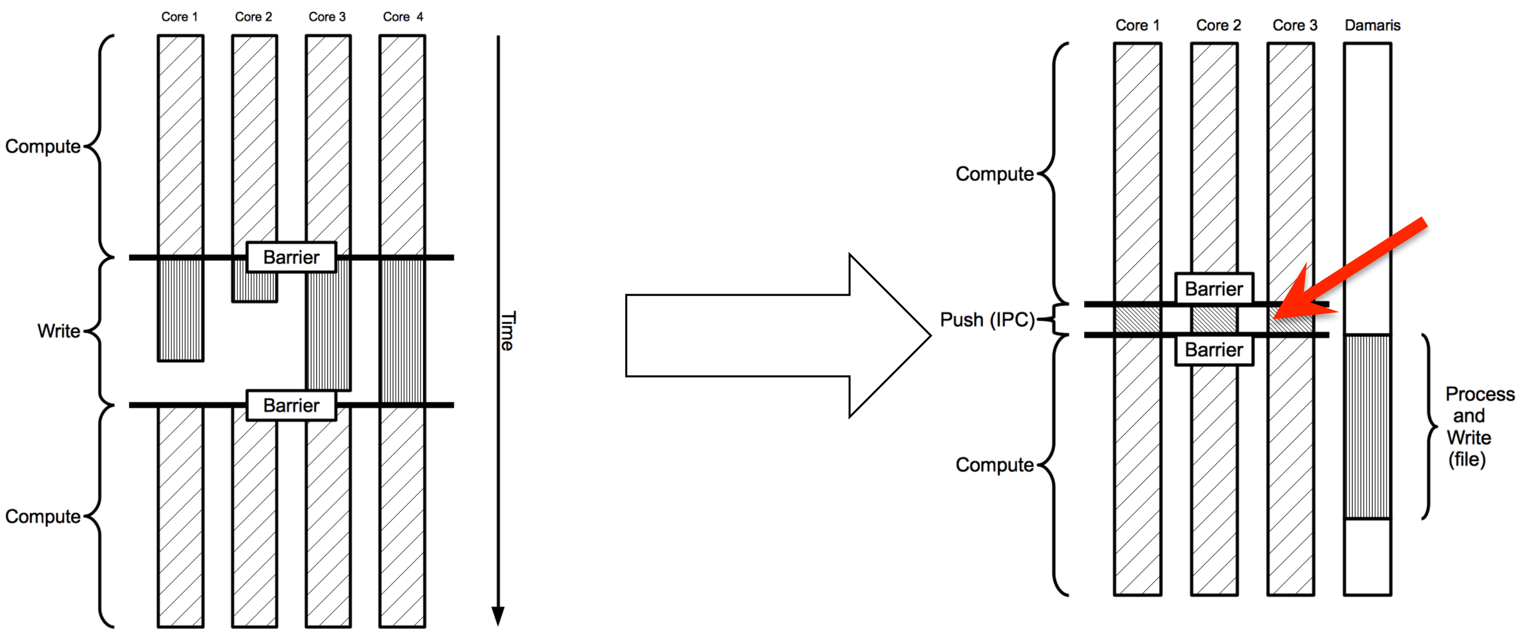
The Damaris approach: dedicated I/O cores



Note: these behaviors are periodic

The Damaris approach: dedicated I/O cores

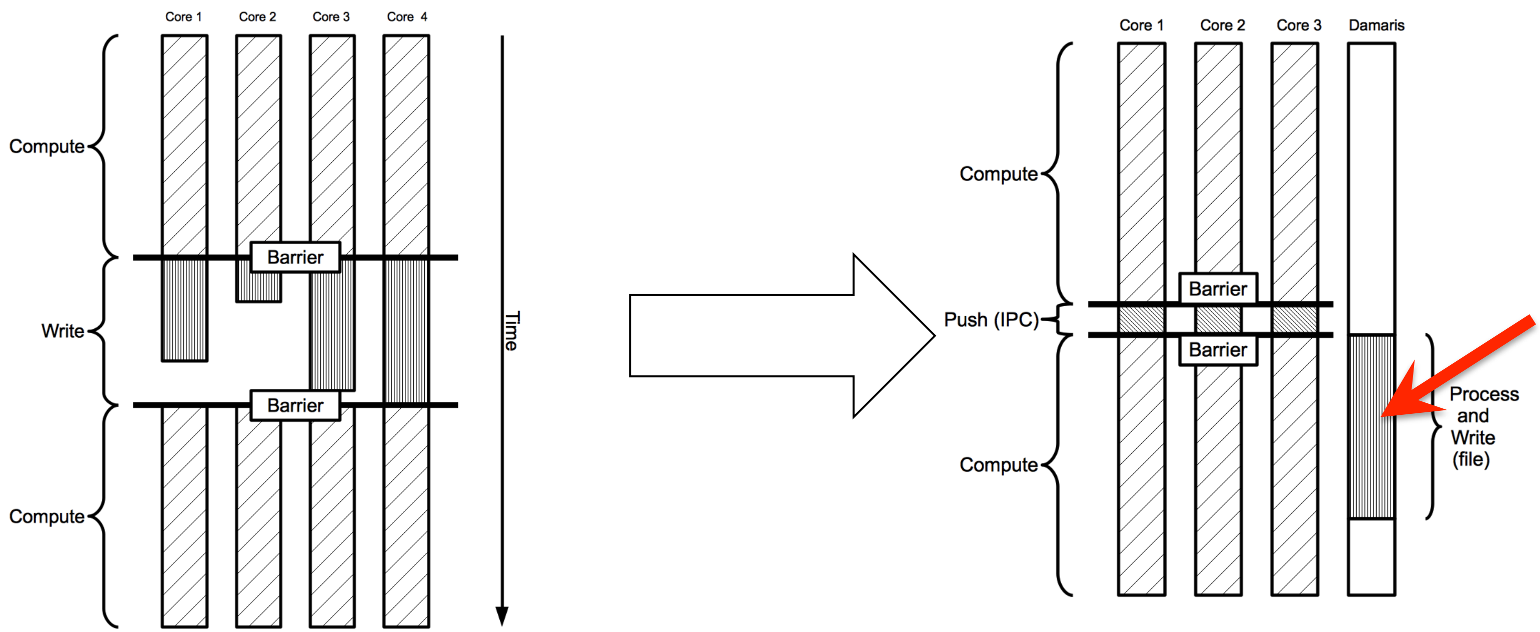
✧ Use the SMP's intra-node shared memory



Note: these behaviors are periodic

The Damaris approach: dedicated I/O cores

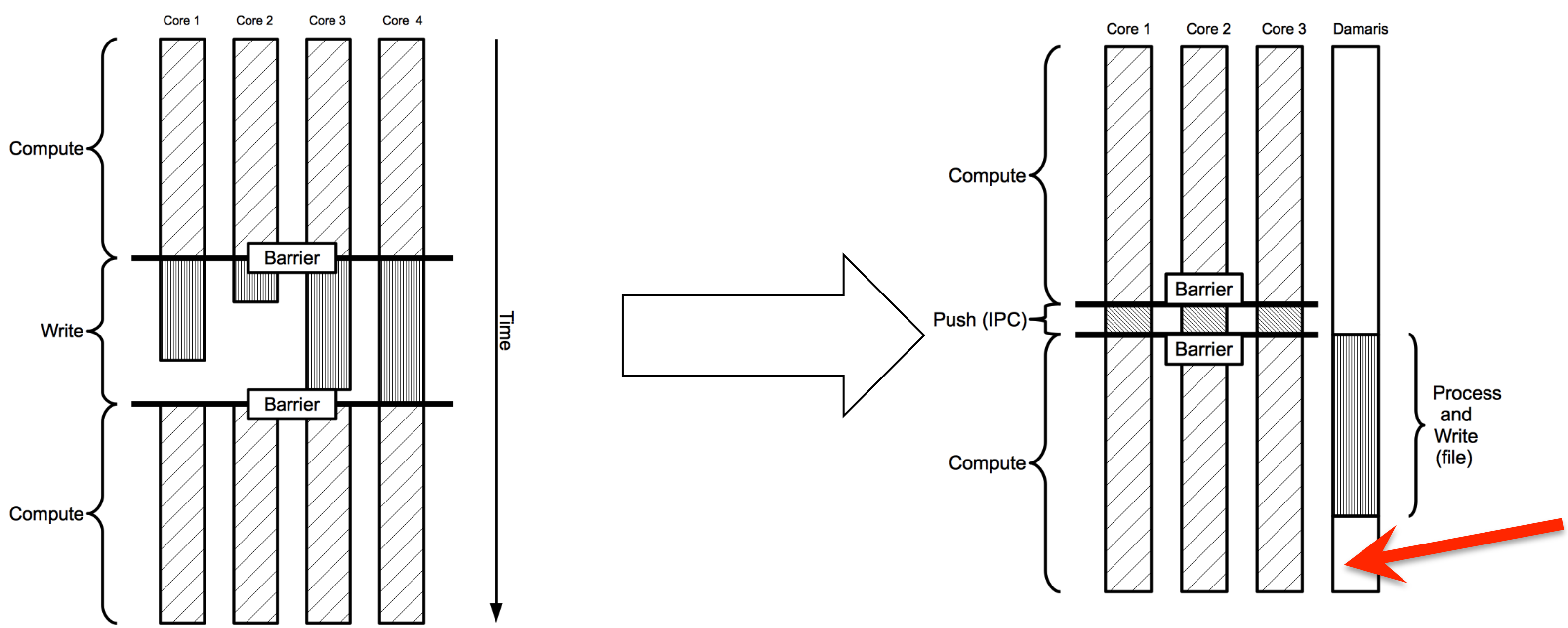
✧ Overlap I/O with computation



Note: these behaviors are periodic

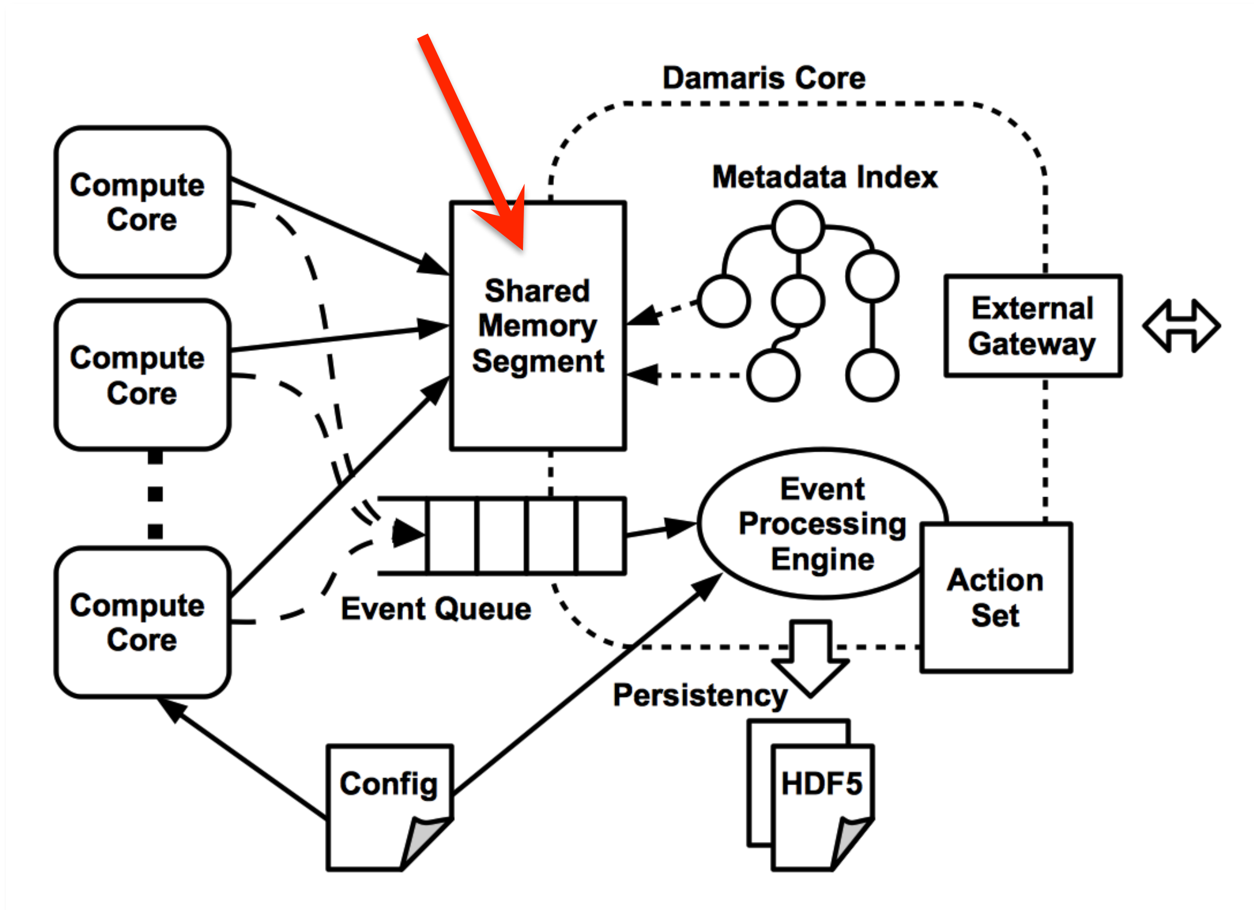
The Damaris approach: dedicated I/O cores

✧ Spare time in the I/O core



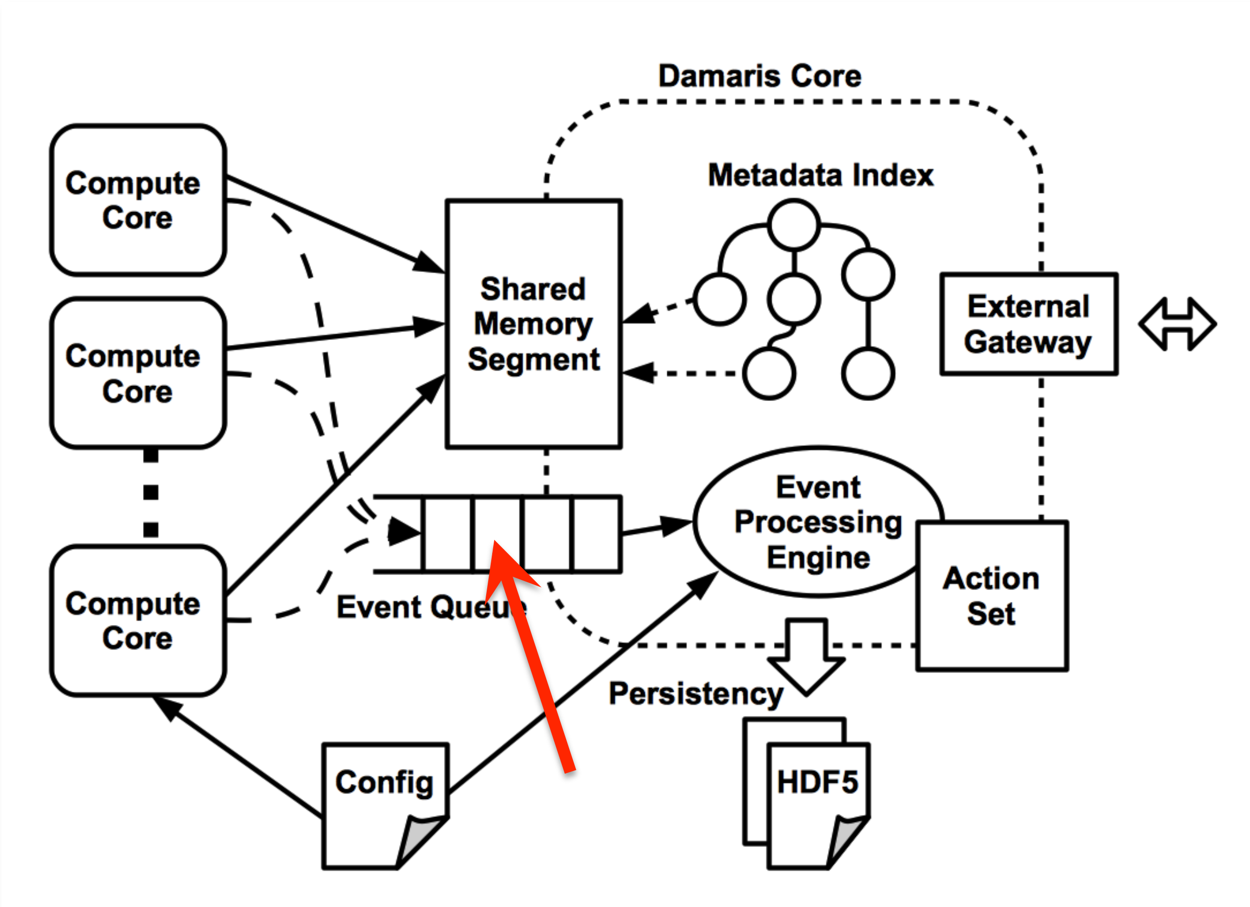
Note: these behaviors are periodic

Damaris: architecture overview



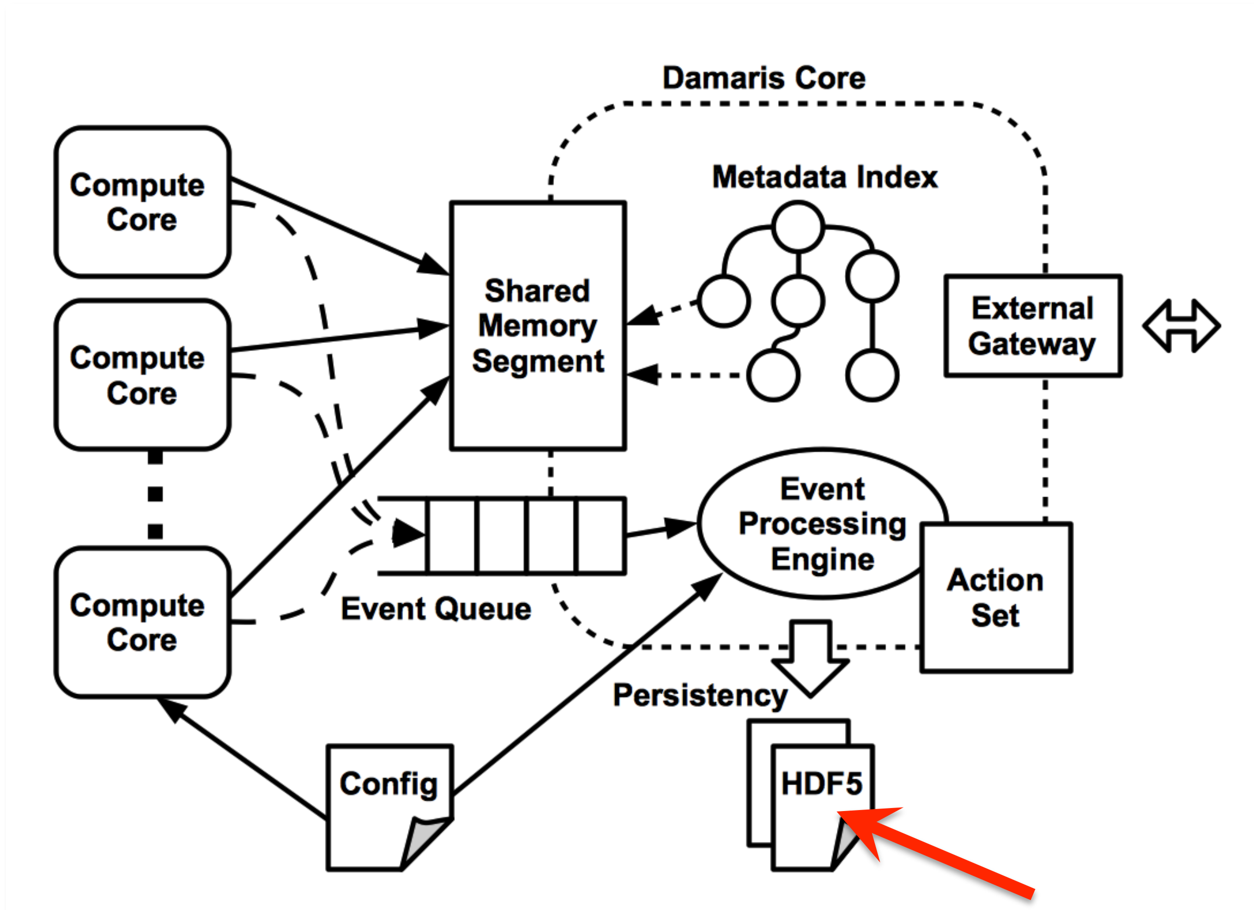
Within one multicore node

Damaris: architecture overview



Within one multicore node

Damaris: architecture overview



Within one multicore node

Damaris: implementation

✧ Written in **C++** (currently 3400 lines of code)

✧ **Boost** library for **interprocess** communications and **shared memory**

✧ Client-side libraries for **C**, **C++** and **Fortran**

✧ External **XML configuration**
(e.g. configuring buffer size, events, actions)

✧ **Take a look! http://damaris.gforge.inria.fr/**

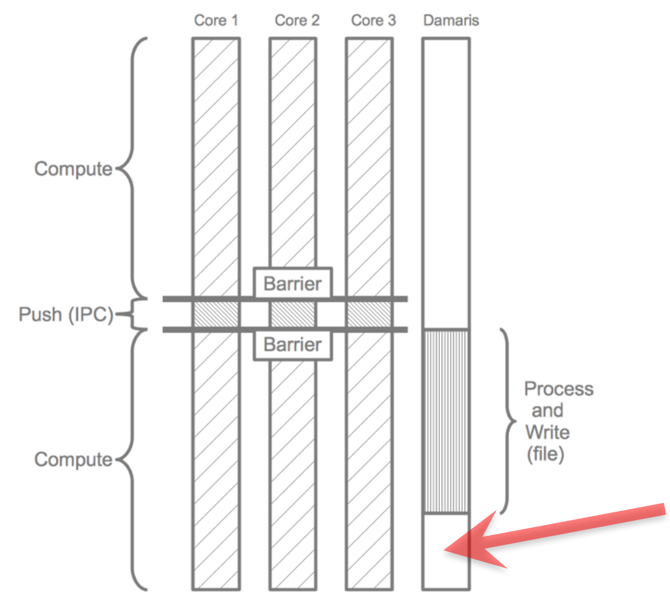
The Damaris approach: benefits

- ✧ Hides I/O-related costs by overlapping computation and I/O
- ✧ Fewer files thanks to data aggregation
- ✧ No synchronization compared to collective I/O
- ✧ No more jitter
- ✧ Spares time

The Damaris approach: benefits

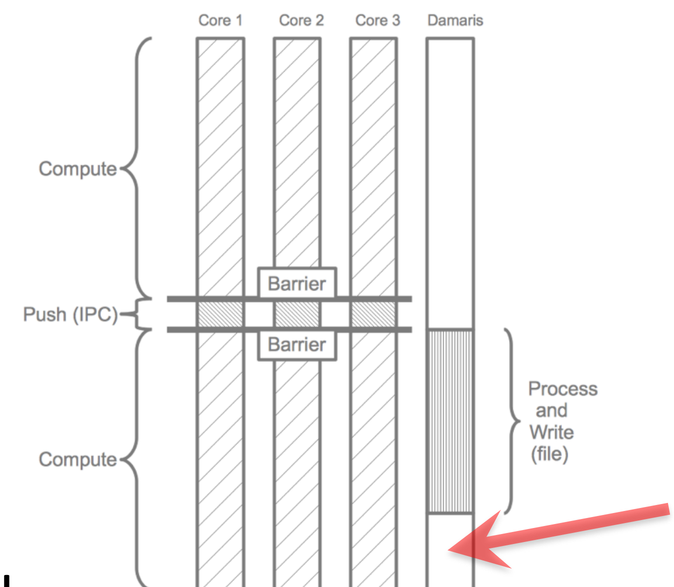
- ✧ Hides I/O-related costs by overlapping computation and I/O
- ✧ Fewer files thanks to data aggregation
- ✧ No synchronization compared to collective I/O
- ✧ No more jitter
- ✧ Spares time

How to use the spare time?



The Damaris approach: benefits

- ✧ Hides I/O-related costs by overlapping computation and I/O
- ✧ Fewer files thanks to data aggregation
- ✧ No synchronization compared to collective I/O
- ✧ No more I/O jitter
- ✧ Spares time



How to use the spare time?

- ✧ **Custom plugin system:**
  - ✧ Data post-processing,
  - ✧ Indexing, analysis
- ✧ **End-to-end scientific process**
  - ✧ Connect visualization/analysis tools
  - ➔ inline visualization

Outline

1. I/O and data management in HPC
2. Understanding I/O jitter
3. Damaris: our new approach to I/O
4. Experimental evaluations
5. Conclusion

The CM1 tornado simulation

- ✧ **CM1 = Georges Bryan's Cloud model version 1**
  - ✧ Three-dimensional, non-hydrostatic, non-linear, time-dependent numerical model suitable for idealized studies of atmospheric phenomena

- ✧ Currently writes using HDF5: **file-per-process**
- ✧ Development version allowing **collective-I/O**

Integration with the CM1 tornado simulation

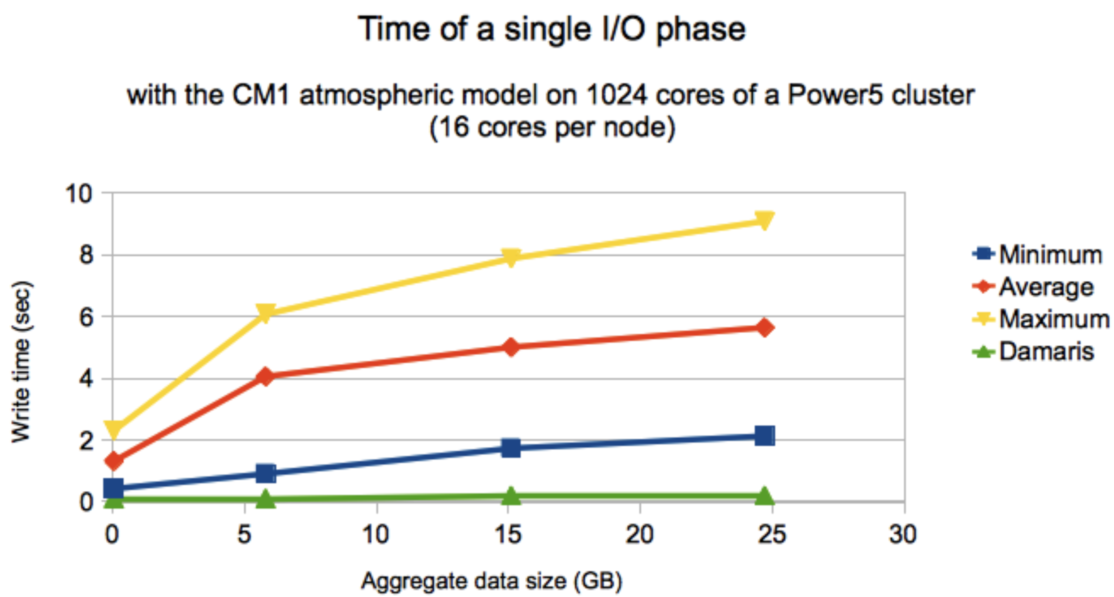
- **On Grid'5000: French national testbed (24 cores/node, 672 cores), with PVFS, comparison with collective I/O**
  - Communication overhead → leaving a core is more efficient
  - No synchronization
  - 6 times higher write throughput

Integration with the CM1 tornado simulation

- **BluePrint: Power5 BlueWaters interim system at NCSA (16 cores/node, 1024 cores), with GPFS, comparison with file-per-process approach**
  - On 64 nodes ➔ 64 files instead of 1024
  - More efficient data aggregation

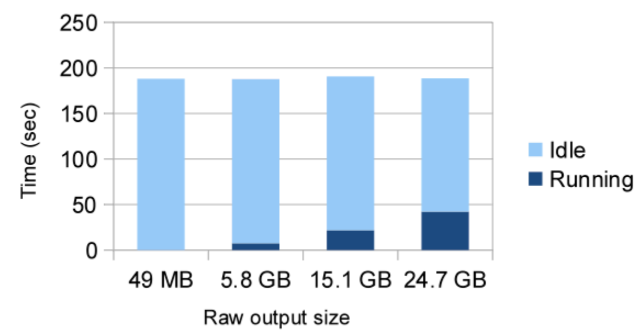
IBM GPFS®

Integration with the CM1 tornado simulation



No more I/O jitter, no more I/O overhead!

Spares more than 75% of time for data processing



Integration with the CM1 tornado simulation

- **In both cases:**
  - Spare time usage
    - Data layout adaptation for subsequent analysis
    - Overhead-free compression (600%)
  - No more I/O jitter and I/O related costs

Outline

1. I/O and data management in HPC
2. Understanding I/O jitter
3. Damaris: our new approach to I/O
4. Experimental evaluations
5. Conclusion

Conclusion

- ✧ **Contribution #1 :** study and representation of I/O variability
  - ✧ Allows simple and fast study of parameters influence

Conclusion

- ✧ **Contribution #1 :** study and representation of I/O variability
  - ✧ Allows simple and fast study of parameters influence

- ✧ **Contribution #2 :** the Damaris approach
  - ✧ Dedicates one core to I/O
  - ✧ Hides I/O variability and overhead
  - ✧ Achieves better throughput (6x on G5k)
  - ✧ Aggregates and compress data (600%)
  - ✧ Tested on Grid’5000 and BluePrint

Outcomes of this work

- ✧ **Poster** presented at **ICS'11** (June 1-3, Tucson, AZ)
    - ✧ **2nd price** at the ICS section of the ACM **Student Research Competition**

Outcomes of this work

- ✧ **Poster** presented at **ICS'11** (June 1-3, Tucson, AZ)
  - ✧ **2nd price** at the ICS section of the ACM **Student Research Competition**

Future work

- ✧ Submission to IPDPS 2012
- ✧ Integrating Damaris in other simulations:
  Enzo, GTC, CESM, WRF…
- ✧ Testing at larger scales *(Kraken)*
- ✧ Using Damaris to perform inline visualization
- ✧ …

People involved in this work:

Dave Semeraro
NCSA, UIUC

Franck Cappello
INRIA Saclay

Gabriel Antoniu
INRIA Rennes

Leigh Orf
CMICH

Marc Snir
NCSA, UIUC

Matthieu Dorier
ENS Cachan

Robert Wilhelmson
NCSA, UIUC

Thank you, questions?

This work is conducted in the context of
the **Joint INRIA/UIUC Laboratory for Petascale Computing**