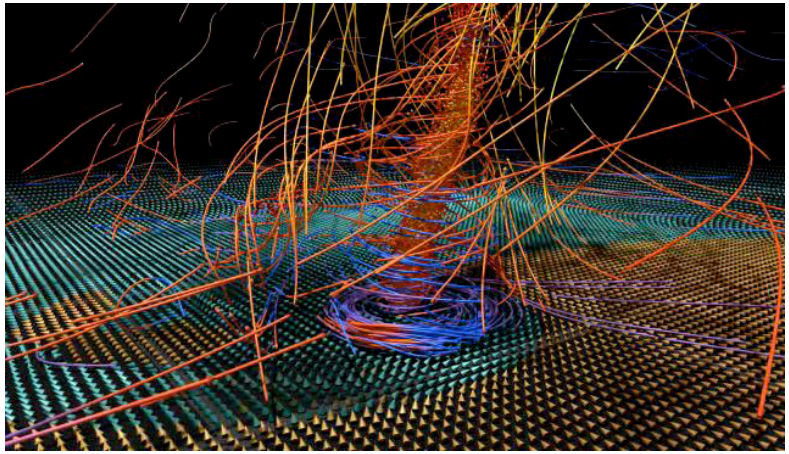
CRIP -- Assises du Big Data
18 Octobre 2012

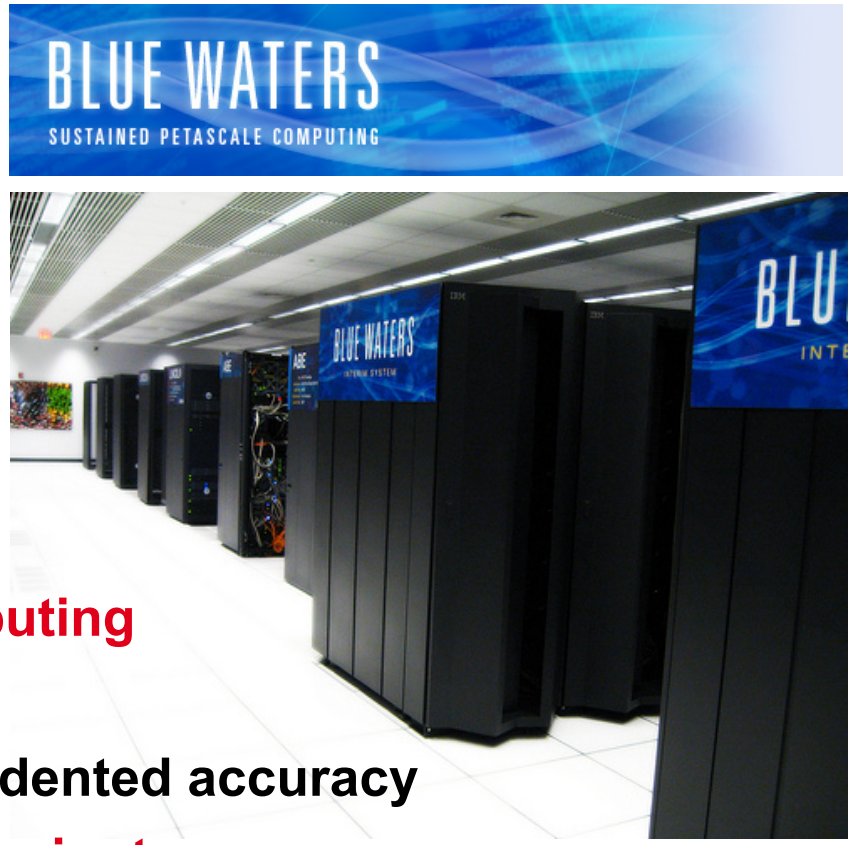
Dealing with Big Data on post-petascale machines: the Damaris approach

Matthieu Dorier
matthieu.dorier@irisa.fr
KerData Team
Inria Rennes, IRISA
ENS Cachan

18 octobre 2012

HPC climate simulations on Blue Waters

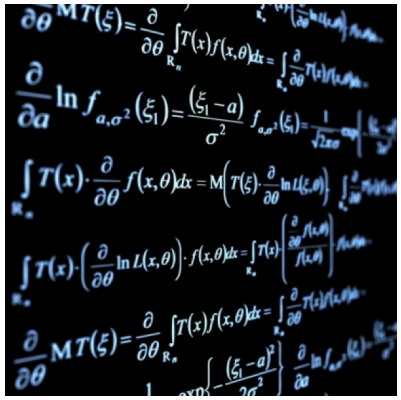


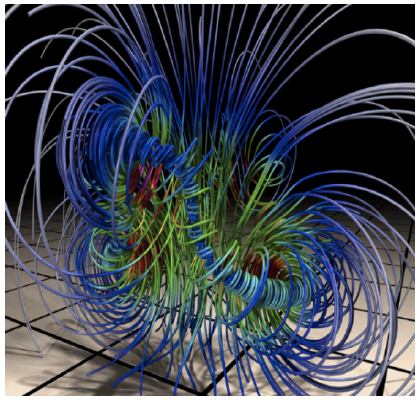


- INRIA – UIUC - ANL

  Joint Lab for Petascale Computing
- Blue Waters: sustained petaflop
- Large-scale simulation at unprecedented accuracy
- Terabytes of data produced every minute

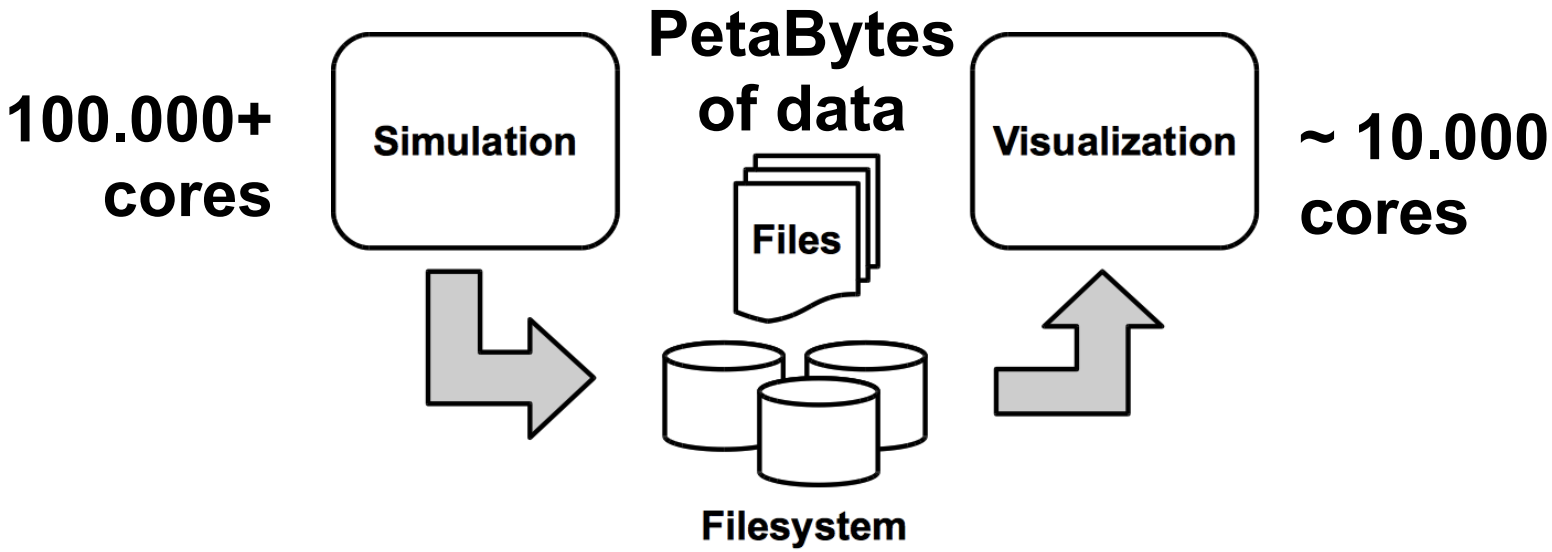
Big Data challenge on post-petascale machines





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

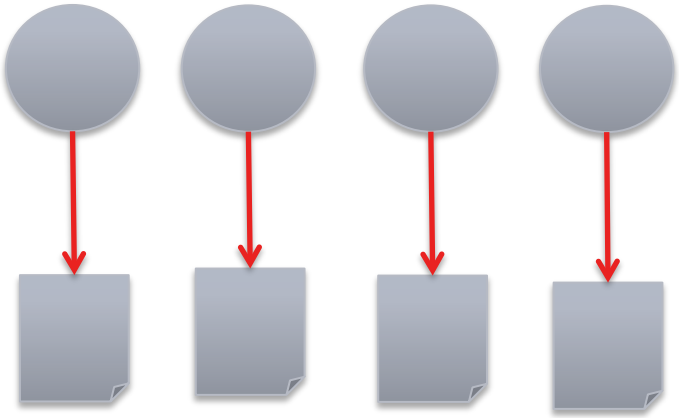
The stardard I/O flow: offline data analysis



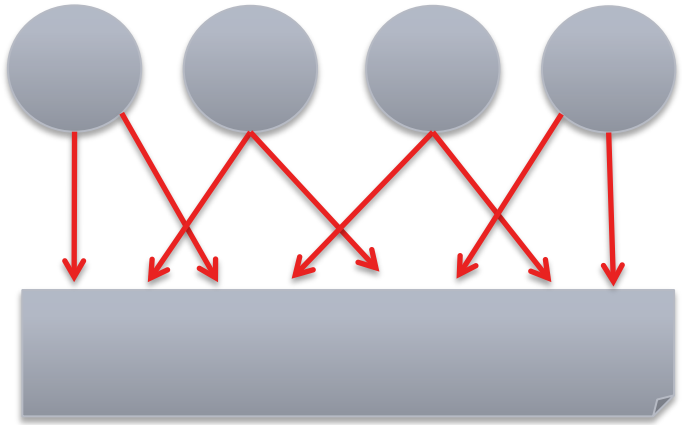
- Periodic data generation from the simulation
- Storage in a parallel file system (Lustre, PVFS, GPFS,…)
- Offline data analysis (on another cluster)

Writing from HPC simulations: File per process vs. Collective I/O

- Two main approaches for I/O in HPC simulations
- Implemented in MPI-I/O, available in HDF5, NetCDF,…



- Too many files
- High metadata overhead
- Hard to read back



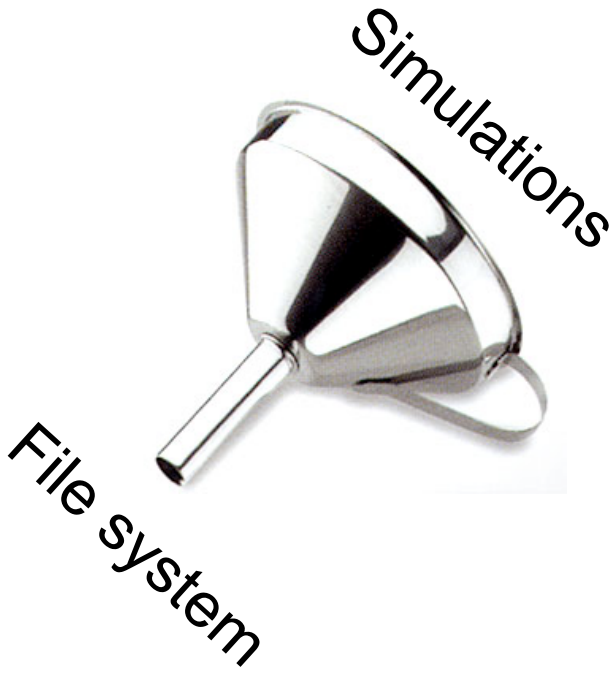
- Requires coordination
- Data communications

Periodic synchronous snapshots lead to I/O bursts and high variability (jitter)

Input
Output

The “cardiogram” of a PVFS data server during a run of the CM1 simulation

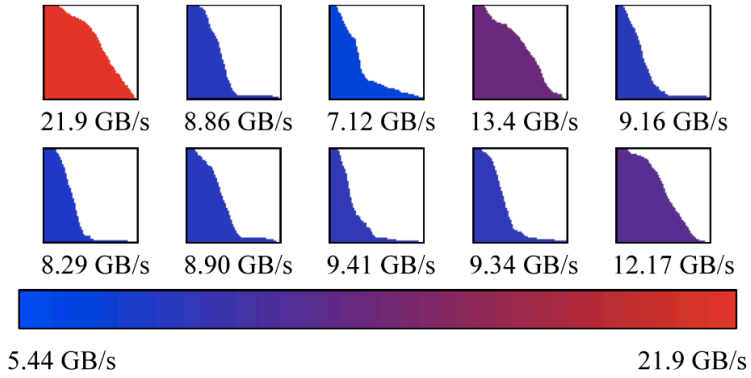
Not only are performances bad, they're also unpredictable!



I/O variability, or "jitter"

- **Write time unpredictability**
  - Between two processes
  - Between two I/O phases
  - Between simulations

- **Due to**
  - Cross-applications interferences
  - Many processes writing at the same time
  - Different amounts of data
  - File system's configuration
  - …



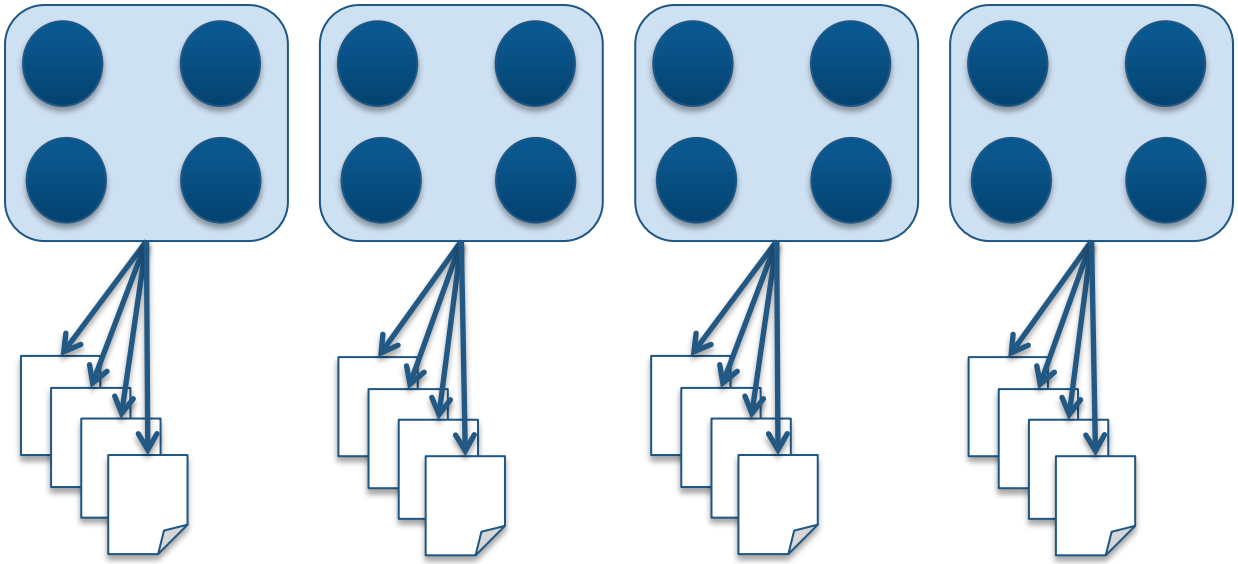
Can we hide achieve scalable, jitter-free I/O?

Yes: Damaris

DAMARIS
“Dedicated Adaptable Middleware for Application Resources Inline Steering”

The Damaris approach: dedicated I/O cores

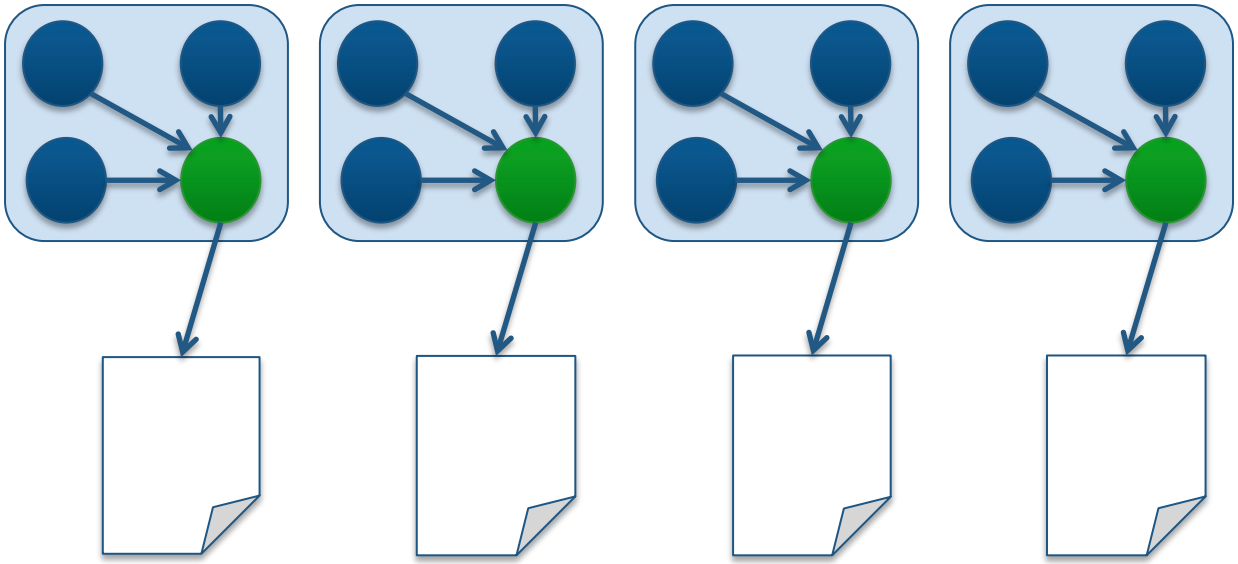
- **Next-generation supercomputers have multicore SMP nodes**



- Network access contention at the level of a node

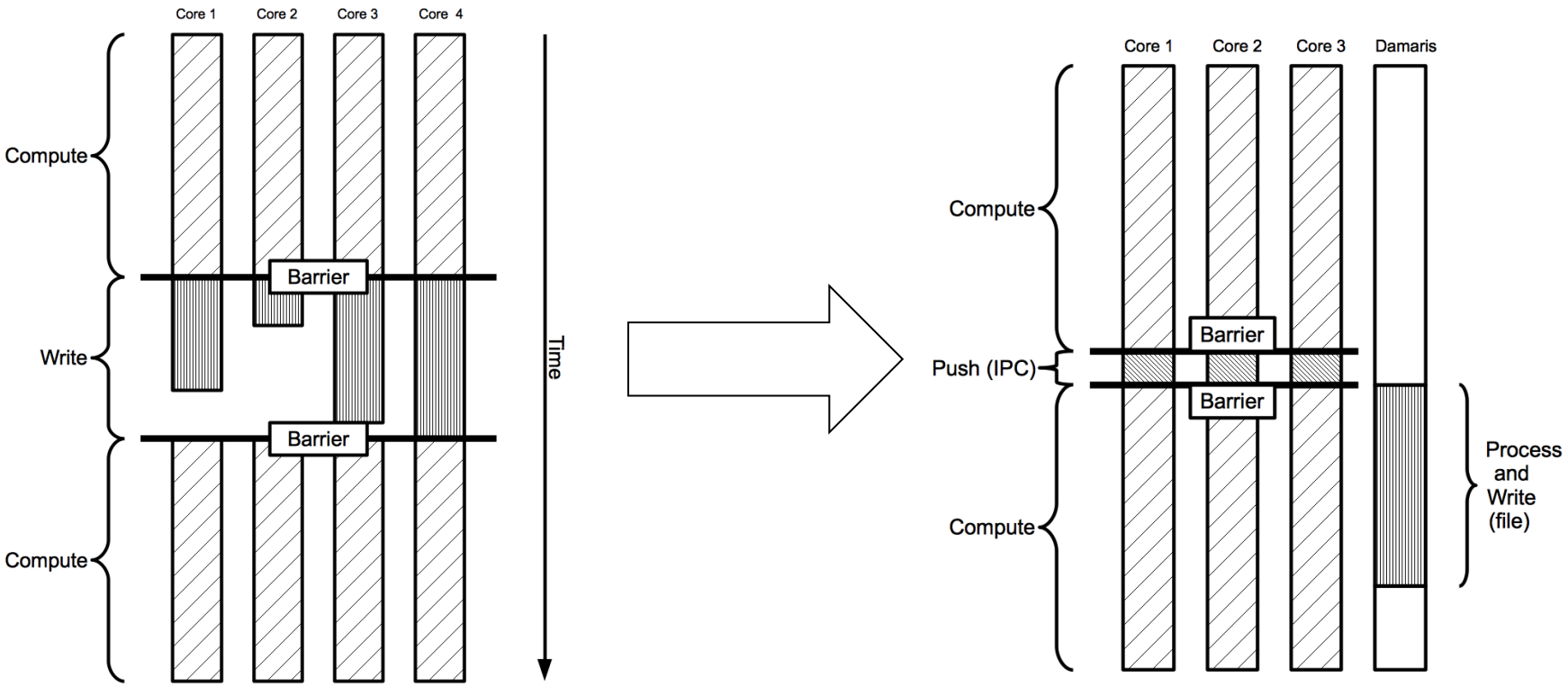
The Damaris approach: dedicated I/O cores

- **Next-generation supercomputers have multicore SMP nodes**



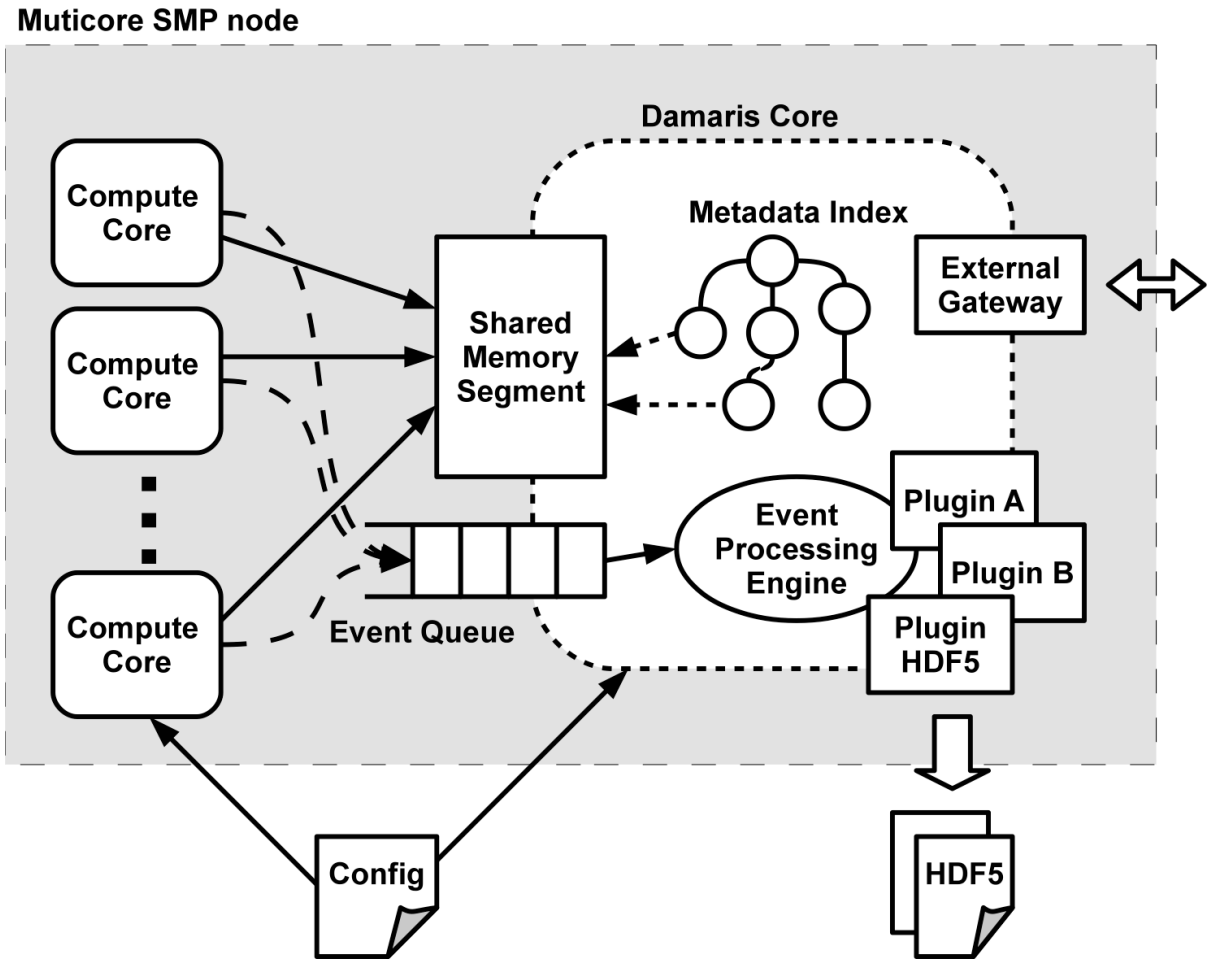
- Network access contention at the level of a node
- Possibility of efficient interactions through shared memory
- One core dedicated for gathering data
- This core only writes

Moving jitter to a dedicated core



Leave a core, go faster!

Damaris at a glance



Damaris: current state of the software

- **Version 0.6** available at http://damaris.gforge.inria.fr/
  - Along with documentation, tutorials and examples
- Written in C++, uses
  - Boost for IPC, Xerces-C and XSD for XML parsing
- API for Fortran, C, C++

- **Tested on**
  - Grid'5000 (Linux Debian), Kraken (Cray XT5 - NICS), JaguarPF (Cray XK6 – Oak Ridge), JYC (Blue Waters testing system: Cray XE6 - NCSA), Intrepid (BlueGene/P – Argonne)
- **Tested with**
  - CM1 (climate), OLAM (climate), GTC (fusion), Nek5000 (CFD)

Experimental evaluation

Running the CM1 simulation on Kraken, G5K and BluePrint with Damaris

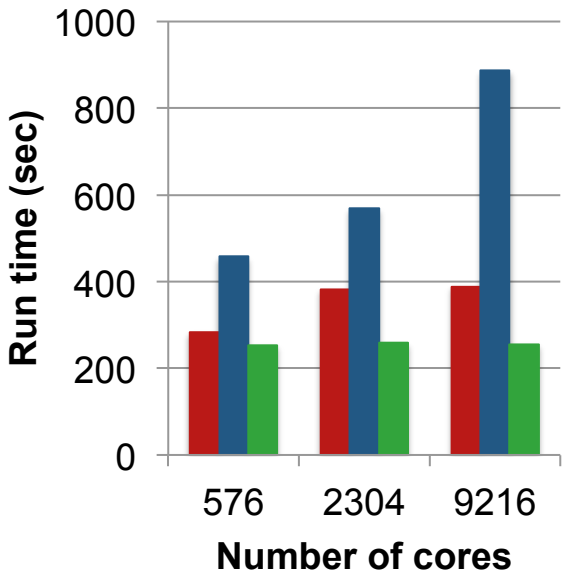
- **The CM1 simulation**
  - Atmospheric simulation
  - One of the Blue Waters target applications
  - Uses HDF5 (file-per-process) and pHDF5 (for collective I/O)

- **Kraken**
  - Cray XT5 at NICS
  - 12 cores/node
  - 16 GB/node
  - Luster file system

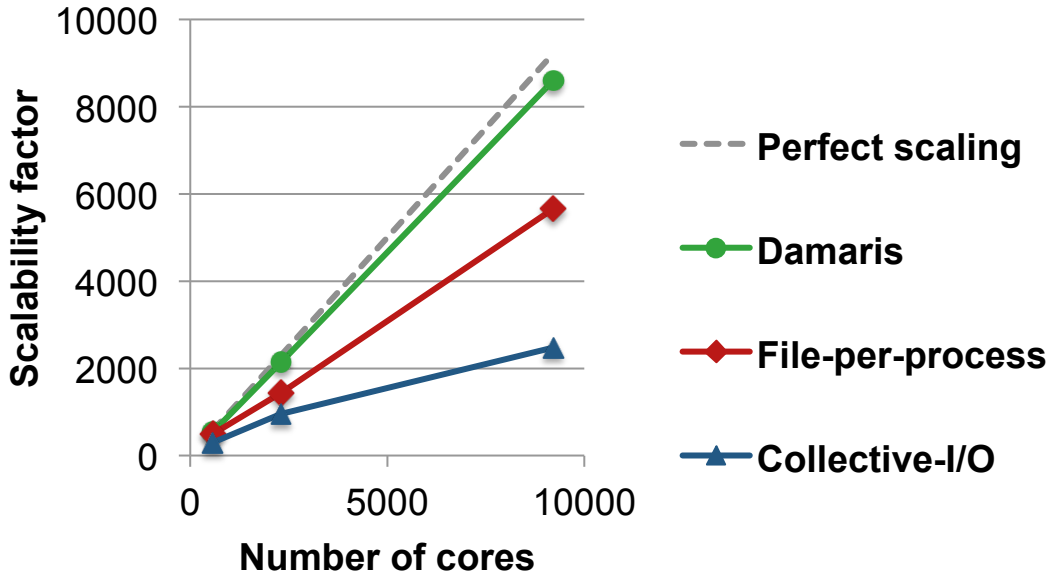
- **Grid 5000**
  - 24 cores/node
  - 48 GB/node
  - PVFS file system

- **BluePrint**
  - Power5
  - 16 cores/node
  - 64 GB/node
  - GPFS file system

Damaris achieves almost perfect scalability



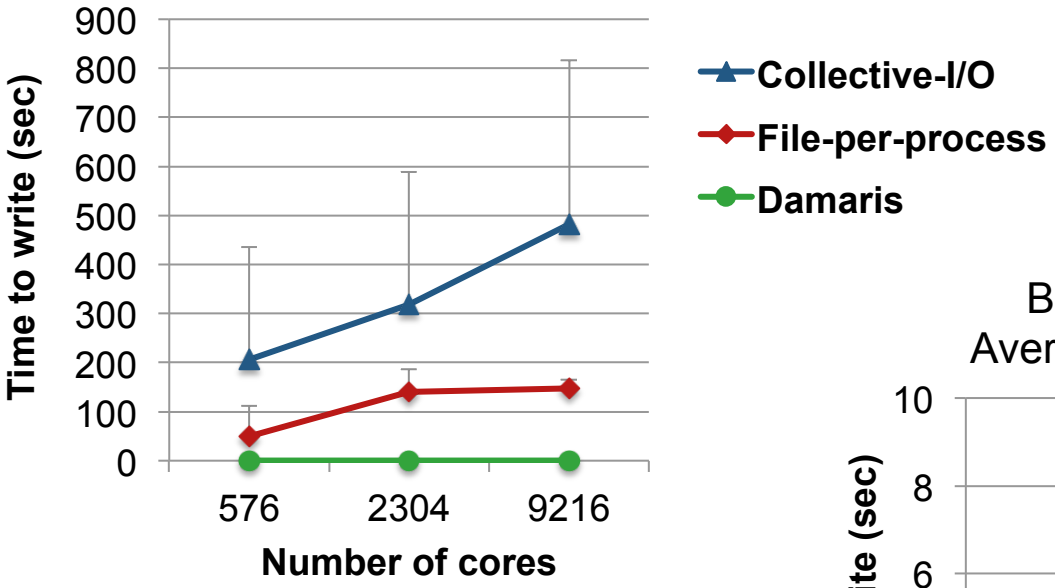
Kraken Cray XT5
Application run time
(50 iterations + 1 write phase)



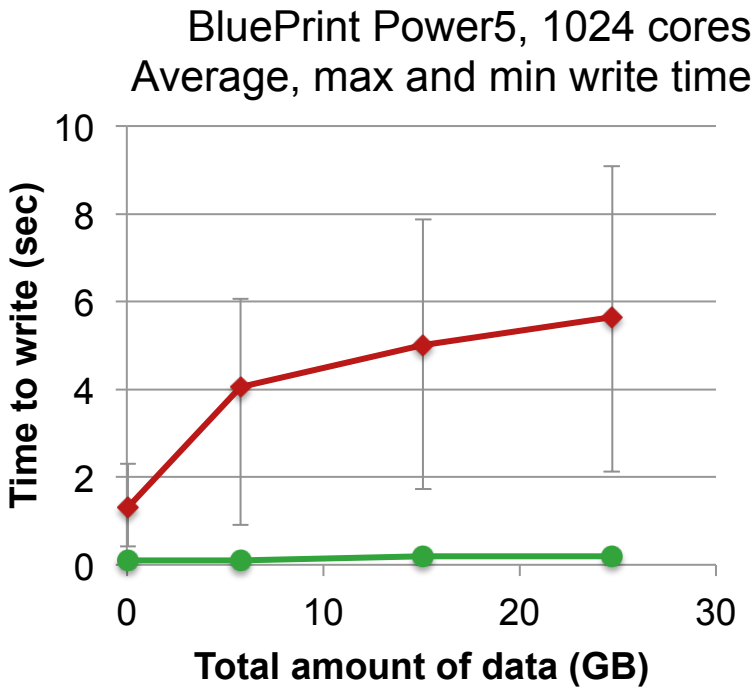
Weak scalability factor $S = N \frac{T_{base}}{T}$

N: number of cores
$\mathbf{T_{base}}$: time of an iteration on one core w/o write
T: time of an iteration + a write

Damaris hides the I/O jitter

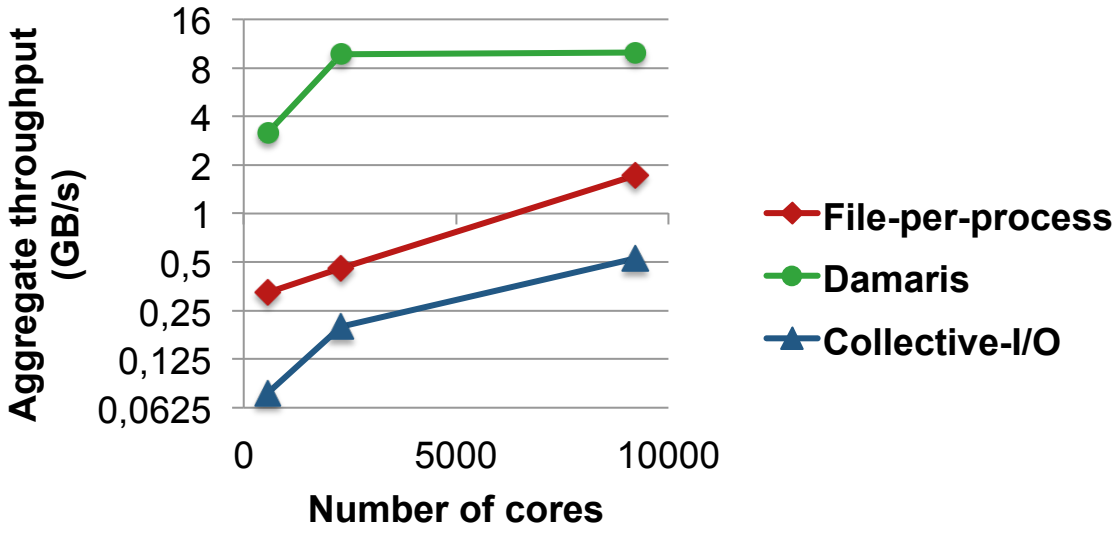


Kraken Cray XT5
Average and maximum write time
28MB per process



Damaris increases effective throughput

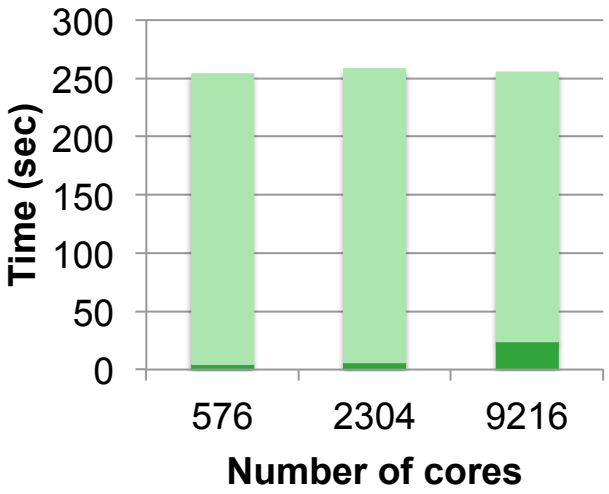
Average aggregate throughput from the writer processes



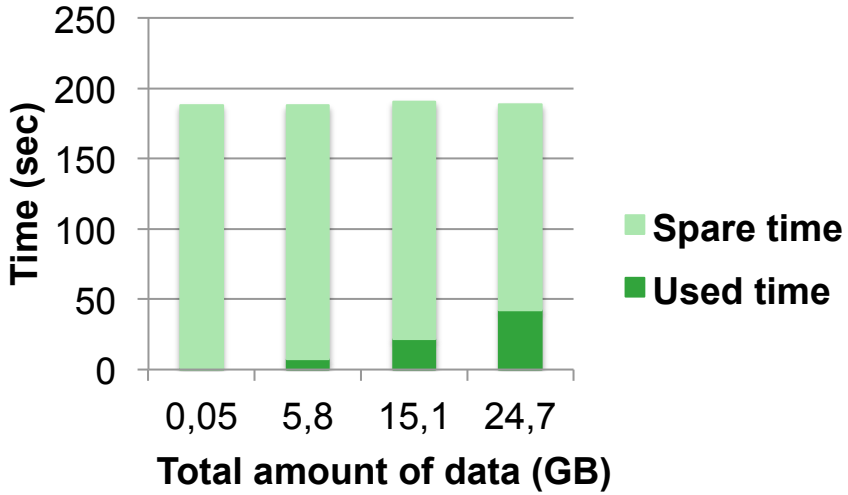
Kraken Cray XT5
Average aggregate throughput from writers

Damaris spares time for data management

Time spent by Damaris writing data and time spent waiting



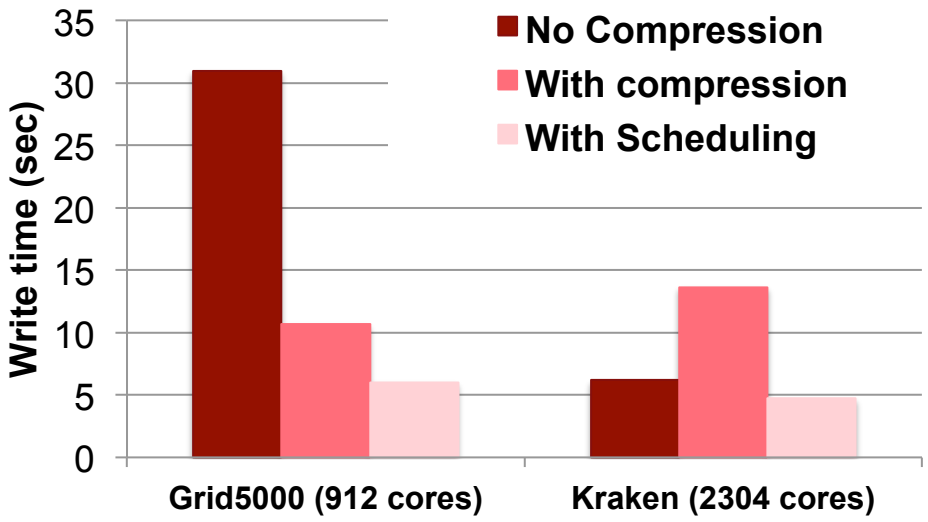
Kraken Cray XT5



BluePrint Power5 (1024 cores)

Damaris spares time?
Let's use it!

Use of the spare time: Adding compression and scheduling



- **With compression enabled: up to 600 percent compression ratio!**
  - (reduction to 16bits float values and lossless gzip level 4)
- **Scheduling**
  - each dedicated core waits for its time slot before writing

Going beyond pure I/O: “in-situ” analysis through Damaris

Towards inline visualization strategies

- **Loosely coupled strategy**
  - Visualization runs on a separate, remote set of resources
  - Partially or fully asynchronous
  - Solutions include staging areas, file format wrappers (HDF5 DSM, ADIOS, …)

- **Tightly coupled strategy**
  - Visualization is collocated with the simulation
  - Synchronous (time-partitioning): the simulation periodically stops
  - Solution by code instrumentation
  - Memory constrained

Four main goals

User friendliness

Low impact on simulation code

Adaptability
(to different simulations and visualization scenarios)

Performance

Low impact on simulation run time

Good resource utilization
(low memory footprint, use of GPU,…)

Driving the acceptance of any approach

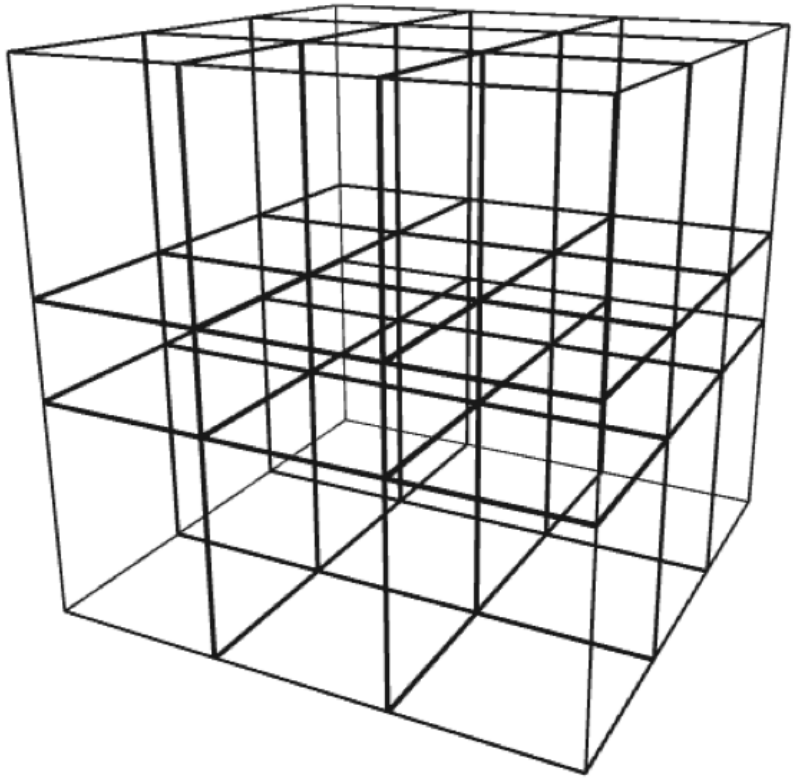
Towards inline visualization strategies

Coupling		Tight	Loose	?
Impact on code		High	Low	Minimal
Adaptability	Interactivity	Yes	None	Yes
	Instrumentation	High	Low	Minimal
Impact on run time		High	Low	Minimal
Resource usage		Good	Non-optimal	Better

- Researchers seldom accept tightly-coupled in-situ visualization
  - Because of development overhead, performance impact…
  - "Users are stupid, greedy, lazy slobs" [1]
- **Is there a solution achieving all these goals?**
- **Yes: Damaris**

[1] D. Thompson, N. Fabian, K. Moreland, L. Ice, "Design issues for performing in-situ analysis of simulation data", Tech. Report, Sandia National Lab

Let's take a representative example



```
// rectilinear grid coordinates
float mesh_x[NX];
float mesh_y[NY];
float mesh_z[NZ];
// temperature field
double temperature[NX][NY][NZ];
```

“Instrumenting” with Damaris

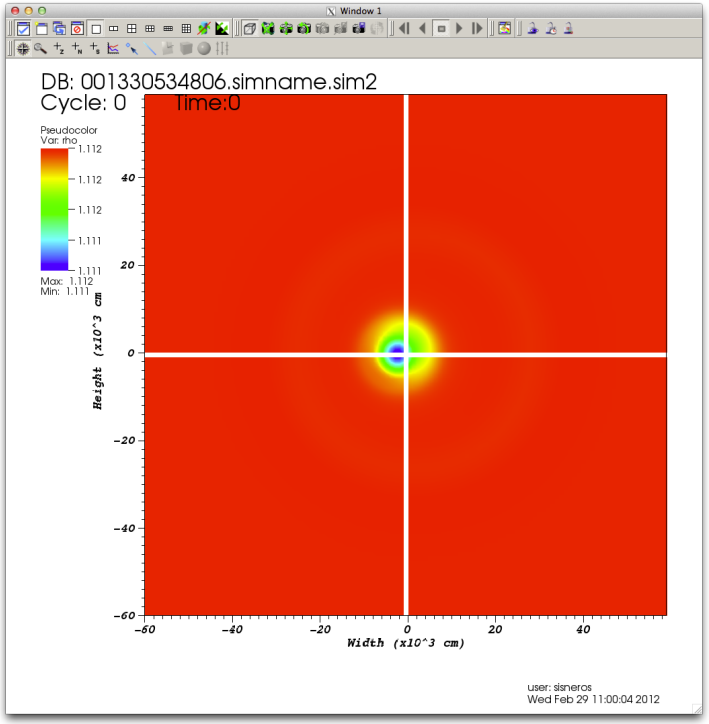
```
DC_write(“mesh_x”,iteration,mesh_x);
DC_write(“mesh_y”,iteration,mesh_x);
DC_write(“mesh_z”,iteration,mesh_x);

DC_write(“temperature”,iteration,temperature);
```

(Yes, that’s all)

Now describe your data in an XML file

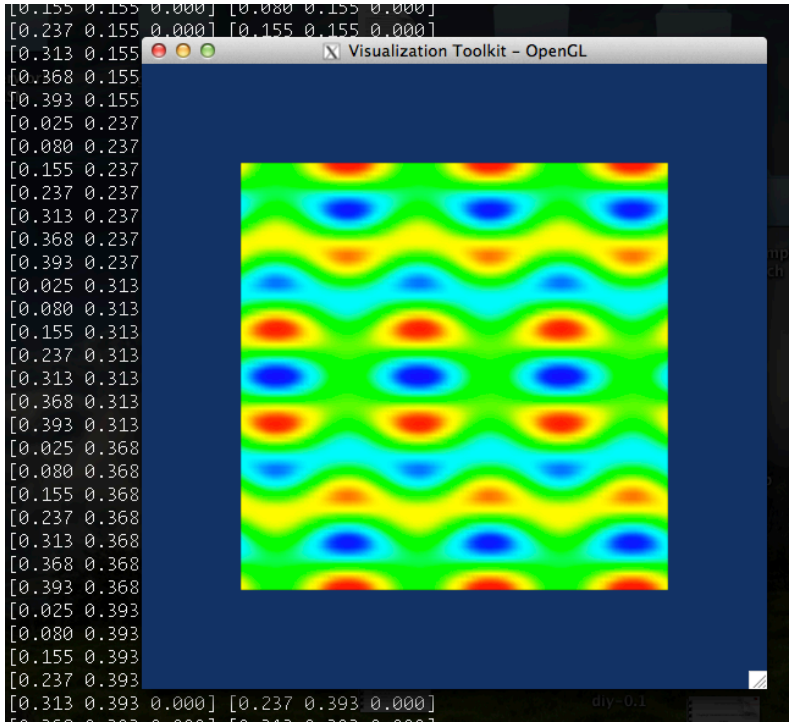
```
<parameter name="NX" type="int" value="4"/>
<layout name="px" type="float" dimensions="NX"/>
<variable name="mesh_x" layout="px">
<!-- idem for PTY and PTZ, py and pz, mesh_y and mesh_z -->

<layout name="data_layout" type="double" dimensions="NX,NY,NZ"/>
<variable name="temperature" layout="data_layout" mesh="my_mesh" />

<mesh type="rectilinear" name="my_mesh" topology="3">
   <coord name="mesh_x" unit="cm" label="width" />
   <coord name="mesh_y" unit="cm" label="depth" />
   <coord name="mesh_z" unit="cm" label="height" />
</mesh>
```

- Unified data description for different visualization software
- Damaris translates this description into the right function calls to any such software (right now: Python and VisIt)
- Damaris handles the interactivity through dedicated cores!



VisIt interacting with the CM1 simulation through Damaris

Plugin exposing vector fields in the Nek5000 simulation with VTK



Conclusion: using Damaris for in-situ analysis

Coupling		Tight	Loose	Damaris
Impact on code		High	Low	Minimal
Adaptability	Interactivity	Yes	None	Yes
	Instrumentation	High	Low	Minimal
Impact on run time		High	Low	Minimal
Resource usage		Good	Non-optimal	Better

- Impact on code: 1 line per object (variable or event)
- Adaptability to multiple visualization software (Python, VisIt, ParaView, etc.)
- Interactivity through VisIt
- Impact on run time: simulation run time independent of visualization
- Resources usage:
  - preserves the “zero-copy” capability of VisIt thanks to shared-memory,
  - can asynchronously use GPU attached to Cray XK6 nodes

Conclusion and future work

- **The Damaris approach**
  - Dedicated cores
  - Shared memory
  - Highly adaptable system thanks to plugins

- **Results**
  - Fully hides the I/O jitter and I/O-related costs
  - 15x sustained write throughput (compared to collective I/O)
  - Almost perfect application scalability
  - Execution time divided by 3.5 compared to collective I/O
  - Overhead-free 600% compression ratio

- **Future work**
  - Efficient coupling of simulation and analysis tools
  - Distributed I/O scheduling on dedicated cores

CRIP -- Assises du Big Data
18 Octobre 2012

Merci !

Dealing with Big Data on post-petascale machines: the Damaris approach

Matthieu Dorier
matthieu.dorier@irisa.fr
KerData Team
Inria Rennes, IRISA
ENS Cachan