From Damaris to CALCioM
Mitigating I/O Interference in HPC Systems

Matthieu Dorier – ENS Rennes, IRISA, Inria Rennes KerData project team
Joint work with Rob Ross, Dries Kimpe, Gabriel Antoniu, Shadi Ibrahim
Within the Data@Exascale associated team

10th workshop of the Joint Lab for Petascale Computing
Urbana-Champaign, November 2013
Outline

• Damaris: After 3 years of collaboration...
• CALCioM: Towards cross-application coordination
Damaris after 3 years...

Originated from the "Shared Buffering System" designed in 2010 during an internship at NCSA, Damaris proposes to dedicate cores in multicore SMP nodes to data management, i.e. storage, in situ analysis and visualization.

Overview

Implementation

- Version 0.7.3 available
- Version 1.0 for summer 2014
- 15095 lines of code
- API for C, C++ and Fortran simulations
- Easy configuration with XML
- In situ visualization with VisIt
- Python and C++ plugins
Damaris after 3 years...

People Involved
Matthieu Dorier, Gabriel Antoniu, Lokman Rahmani, Roberto Sisneros, Dave Semeraro, Bob Wilhelmson, Rob Ross, Tom Peterka, Dries Kimpe, Marc Snir, Franck Cappello, Leigh Orf

Publications


M. Dorier, advised by G. Antoniu. Efficient I/O using Dedicated Cores in Large-Scale HPC Simulations. PhD forum of IPDPS 2013


Evaluated on
Blue Waters, Intrepid, Kraken, Jaguar, Grid’5000, Blue Print, Surveyor

Evaluated with
CM1, Nek5000, OLAM
Mitigating I/O Interference in HPC Systems
Introduction to cross-application interference

**Interference:** Performance degradation observed by an application in contention with other applications for the access to a shared resource.

- How often does I/O interference occur?
- What is the effect of I/O interference?
- How do we quantify and visualize it?
- How to mitigate it?
How often does I/O interference occur?
How often does interference occur?

“Intrepid has a really weird workload compared to most other systems, because of the large number of large jobs.”

Narayan Desai (ANL)
How often do interference occur?

I am an application, I start writing, what is the probability that at least one other application is also accessing the file system?

\[
P(\text{another is doing I/O}) = 1 - \sum_{n=0}^{+\infty} P(X = n)(1 - E(\mu))
\]

Where \(X\) is the number of running application (random variable), \(\mu\) is the I/O time v.s. computation time ratio of applications (r.v.), Assuming independence between \(X\) and \(\mu\).

On Intrepid:
Assuming \(E(\mu) = 5\%\), \(P(\text{another is doing I/O}) = 64\%\)
What is the effect of I/O interference?
What is the effect of I/O interference?

IOR running on 336 cores, writing every 10 seconds in a 35-server PVFS file system on Grid’5000

A second instance is started on 336 other cores, writing the same amount of data every 7 seconds

I/O interference has a large impact on caching mechanisms
How do we quantify and visualize I/O interference?
Interference factor

• The user is interested in the *factor by which interference increases the I/O time*:

\[
I_X = \frac{T_X}{T_X(\text{alone})} > 1
\]

• Considering \( n \) applications, we could (for example) want to minimize the *sum* of access times:

\[
f = \sum_{X \in \text{app}} T_X
\]

• These metrics can be adapted to anything (Energy consumption, CPU cycles, etc.): \( f \) can be generalized as a metrics for machine-wide efficiency.
Results on Surveyor (2x 2048 cores), each core writes 8MB contiguously. The graph represents the point of view of one of the 2 applications.
Bad luck for small applications

Experiment on Grid’5000, App B on 24 cores, App A on 744, writing 8MB per process

Smallest App observes an up to 14x decrease of performance! Biggest one does not even see it!
How to mitigate I/O interference?
The CALCioM approach
The CALCioM architecture

Cross-Application Layer for Coordinated I/O Management
CALCioM’s API

CALCioM_Init(MPI_Comm c)
CALCioM_Prepare(MPI_Comm c, MPI_Info i)
CALCioM_Ask()
CALCioM_Check(int* status)
CALCioM_Wait()
CALCioM_Release()
CALCioM_Complete()
CALCioM_Finalize()
Possible coordination strategies

“First come first served” (FCFS) Serialization

```
App A
```

![Diagram](image)

```
App B
```

Interruption

```
App A
```

![Diagram](image)

```
App B
```

```
App A
```

```
App B
```

```
dt
```

```
dt
```
How to choose a coordination strategy

**Q:** Given application A with expected access time $T_A$ and application B with expected access time $T_B$, starting $dt$ time units after application A’s access,

Should A be interrupted in favor of B?
Or should B wait for A to terminate its access?

**Example:** if neither A nor B have something else to do, optimizing global performance, i.e. minimizing an interference effect given by

$$ f = \frac{T_A}{T_A(\text{alone})} + \frac{T_B}{T_B(\text{alone})} $$

Tells us that B should interrupt A if and only if

$$ dt < \frac{T_A^2}{T_A(\text{alone})} - \frac{T_B^2}{T_B(\text{alone})} $$

$$ f = T_A + T_B $$

$$ dt < T_A(\text{alone}) - T_B(\text{alone}) $$
Integration in Mpich

- MPI_Init and MPI_Finalize overwritten in libcalciom.a
- MPI_File_open("myfile")
  - MPI_File_open("calciom:myfile")
- MPI_File_open("pvfs2:myfile")
  - MPI_File_open("calciom:pvfs2:myfile")
- Connection between applications: could be done through MPI_Comm_connect/accept (ideally would benefit from MPI_Comm_iconnect/iaccept) + interaction with the job scheduler
Experimental evaluation
Example of application

2x 2048 cores on Surveyor

- **App A**: 4 files, 4 MB per file per process, contiguous layout
- **App B**: 1 file, 4 MB per file per process, contiguous layout

\[ f = T_A + T_B \]

\[ dt < T_{A\text{ (alone)}} - T_{B\text{ (alone)}} \]
Example of application

App B (small I/O load)  App A (big I/O load)

App B arrives first, App A is serialized after B
Example of application

App B arrives during the write of the 3 first files of App A, Condition indicates that A should be interrupted. The level of interruption produces different patterns.
Example of application

App B arrives during the last write of App A. Condition dictates that B is serialized after A.
Synthesis

CALCioM manages to improve the computational efficiency of the set of applications by avoiding interference, and thus improves the efficiency of the entire machine.
Conclusion
Conclusion

• **Interference between application impacts system efficiency**

• **CALCioM:**
  • Communication layer between independent applications
  • Cross-application coordination through exchange of knowledge on I/O patterns
  • Several policies implemented: FCFS, interruption

Thank you!
Questions?