Towards Scalable Data Management for Map-Reduce-based Data-Intensive Applications on Cloud and Hybrid Infrastructures

Frédéric Suter
Joint work with Gabriel Antoniu, Julien Bigot, Cristophe Blanchet, Luc Bougé, François Briant, Franck Cappello, Alexandru Costan, Frédéric Desprez, Gilles Fedak, Sylvain Gault, Kate Keahey, Bogdan Nicolae, Christian Pérez, Anthony Simonet, Bing Tang & Raphael Terreux
Here Comes the Data Deluge

Experiments  Simulations  Archives  Literature  Instruments

Petabytes
Doubling every 2 years

Credits: Microsoft
Some Challenges for Scalable Storage

Data:
• Massive, unstructured data objects (Terabytes)
• Many data objects ($10^3$-$10^6$)
• High concurrency ($10^3$ concurrent clients)
• Fine-grain access (Megabytes)

Applications:
• Map-Reduce-based data-analysis applications
• Governmental and commercial statistics
• Data-intensive HPC simulations
• Checkpointing for massively parallel computations

Platforms:
• large clusters, clouds, HPC machines, hybrid infrastructures
What is MapReduce?

A simple programming model for data-intensive computing

- Typical problem solved by MapReduce
  - Read a lot of data
  - **Map**: extract something you care about from each record
  - Shuffle and Sort
  - **Reduce**: aggregate, summarize, filter, or transform
  - Write the results

- Approach: *hide messy details* in a runtime library
  - Automatic parallelization
  - Load balancing
  - Network and disk transfer optimization
  - Transparent handling of machine failures

- Implementations: Google MapReduce, Hadoop (Yahoo!)
  - Today Hadoop is *everywhere*!
MapReduce: Pros and Cons

Progress

- Simple programming model
- Scalability to a certain extent

Limitations

- Low throughput for massively concurrent accesses
- Limited object size for cloud object storage (S3, etc.)
- Fault tolerance is rudimentary
- Scheduling: there is room for progress
- Hybrid platforms not really explored yet
Hybrid platforms: why?

• Reason 1:
  • Many cloud providers, many offers
  • Goal: be able to combine various IaaS-level offers and switch between offers while keeping the same data analysis platform

• Reason 2:
  • Rely on (free!) internal compute resources (Enterprise Desktop Grids) and extend to the cloud when needed
  • Goal: minimize costs
Context: 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 (INRIA, Rennes) – leader
- INRIA - AVALON team (Lyon), France
- Nimbus team, 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
Application Case Study: Protein Structure Analysis with SuMo

- **Proteins** are major components of life
  - 3D structure of a protein is essential
  - +70,000 structures in a unique database (Protein Data Bank)
  - A lot more models!
- **SuMo** - Surf the Molecules – initially developed in CNRS IBCP
  - Perform coarse-grain pairwise comparisons of protein structures
  - Compare a set of structures against a database (10^2 GB data)
- **Use case**: find cross reaction allergy of arachide “Ara H1” allergen
  - reference database of 3,541 models, from 6,000 sequences
  - Previous experiments were running during 10s of hours
- **Goal**
  - Analyze large datasets with the SuMo method
  - Take advantage of Map Reduce optimizations
Global Architecture and Associated Goals

Focus on the data storage and management
New scheduling techniques for large executions of Map-Reduce instances
Fault tolerance and security
Outline

• Introduction
• Proposed Storage Architecture
  • Clouds: BlobSeer on Nimbus-powered clouds
  • Desktop Grids: BitDew
• Scheduling issues
• Fault tolerance issues
• Combining everything
• Conclusions
MapReduce on Clouds: The BlobSeer Approach to Concurrency-Optimized Data Management

BlobSeer: software platform for scalable, distributed BLOB management
- Decentralized data storage
- Decentralized metadata management
- Versioning-based concurrency control
- Lock-free concurrent writes (enabled by versioning)

A back-end for higher-level data management systems
- Short term: highly scalable distributed file systems
- Middle term: storage for cloud services
- Long term: extremely large distributed databases

http://blobseer.gforge.inria.fr/
Highlight: BlobSeer does better than Hadoop!

MapReduce: a natural application class for BlobSeer

MapReduce for Desktop Grid: the BitDew Approach

• Issues specific to Desktop Grids
  - No shared file system nor direct communication
  - Fault and host churns

• Solutions
  - Data replication management
  - Result certification of intermediate data
  - Collective operation (scatter + gather/reduction)

• Optimized runtime
  - Latency hiding, barrier-free computation, 2-level scheduler, ...
Highlight: Bitdew is scalable

- WordCount benchmark
  - 512 nodes
  - Size of the document doubles with number (2TB)
Scheduling Issues and Proposal

• Several places where scheduling is needed
  • Partitionning and balancing data
  • Map ↔ Reduce transfer
• Start from a work from Berlinska and Drozdowsk
  • Choose the amount of data per node
  • Determine how to rebalance data
  • Order transfers between Map and Reduce tasks
• Contributions
  • Simplify the linear program into a linear system
  • Schedule transfers as soon as possible (15-20% gain on transfer time)
  • Allow preemption by high priority transfers
Highlight: Execution Time Reduction
Dealing with fault tolerance

• MapReduce follows a simple fault tolerance model:
  • Wait a predefined amount of time for a mapper to complete
  • If the mapper didn't finish in time, assume it has failed and re-execute it
  • Works well for short-lived mappers
  • However, it is not enough for long-lived mappers or complex mappers that are instances of tightly coupled applications

• Proposal: BlobCR
  • Target platform: IaaS clouds (i.e. assumes mapper runs inside virtual machines)
  • Key idea: capture the state of the virtual disks
  • Create a globally consistent snapshot from these virtual disks
  • Store the global snapshot persistently into a dedicated repository
  • Restart from the global snapshot in case of failures to minimize amount of lost computation

• How to use it?
  • Magic primitive inside the mapper called CHECKPOINT FS
  • Can be called explicitly or triggered
Combine Everything with Components

- High Level Component Model (INRIA-Avalon)
  - Hierarchical component model, connector based
  - Generic model (template à la C++)
    - Enable to define MapReduce skeletons!
  - Abstract model (primitive component and connectors)

- A HLCM application is transformed into an executable application
  - Take into account targeted platform (BlobSeer, Bitdew, both, etc.)
  - Insert optimizations (partitioning algorithm are not hard coded in the platform)

- HLCM/L2C: a specialization of HLCM
  - Primitive components in C++ (Java or Python can be added)
  - Primitive connectors: C++ method invocation, CORBA, MPI
    - More primitive connectors are easy to add: WebService, …

- HLCM provides a framework to build an hybrid MapReduce platform
Conclusion

- MapReduce: many issues have not been solved yet!
- Open problems
  - High-throughput data-intensive processing
  - Fault tolerance
  - Optimized scheduling
- Our approach: an integrated architecture for hybrid platforms leveraging efficient bricks: Nimbus, BlobSeer, BitDew
- Future work: integrate and experiment our global architecture on the Grid’5000 testbed, FutureGrid and OpenCirrus
Towards Scalable Data Management for Map-Reduce-based Data-Intensive Applications on Cloud and Hybrid Infrastructures

Thank you!
MapReduce: Pros and Cons

Progress
- Simple programming model
- Scalability to a certain extent

Limitations
- Low throughput for massively concurrent accesses
- Limited object size for cloud object storage (S3, etc.)
- Fault tolerance is rudimentary
- Scheduling: there is room for progress
- Hybrid platforms not really explored yet
Hybrid platforms: why?

• Reason 1:
  • Many cloud providers, many offers
  • Goal: be able to combine various IaaS-level offers and switch between offers while keeping the same data analysis platform

• Reason 2:
  • Rely on (free!) internal compute resources (Enterprise Desktop Grids) and extend to the cloud when needed
  • Goal: minimize costs
Strategic goals:

- Reading
- Writing

 Baghdad

Sage

Stargal
MapReduce on Clouds: The BlobSeer Approach to Concurrency-Optimized Data Management

BlobSeer: software platform for scalable, distributed BLOB management
- Decentralized data storage
- Decentralized metadata management
- Versioning-based concurrency control
- Lock-free concurrent writes (enabled by versioning)

A back-end for higher-level data management systems
- Short term: highly scalable distributed file systems
- Middle term: storage for cloud services
- Long term: extremely large distributed databases

http://blobseer.gforge.inria.fr/
MapReduce for Desktop Grid: the BitDew Approach

• Issues specific to Desktop Grids
  - No shared file system nor direct communication
  - Fault and host churns

• Solutions
  - Data replication management
  - Result certification of intermediate data
  - Collective operation (scatter + gather/reduction)

• Optimized runtime
  - Latency hiding, barrier-free computation, 2-level scheduler, ...
Highlight: Bitdew is scalable

- WordCount benchmark
  - 512 nodes
  - Size of the document doubles with number (2TB)

![Graph showing throughput (MB/s) vs number of nodes (4, 8, 16, 32, 64, 128, 256, 512)]
Scheduling Issues and Proposal

• Several places where scheduling is needed
  • Partitionning and balancing data
  • Map ↔ Reduce transfer
• Start from a work from Berlinska and Drozdowsk
  • Choose the amount of data per node
  • Determine how to rebalance data
  • Order transfers between Map and Reduce tasks
• Contributions
  • Simplify the linear program into a linear system
  • Schedule transfers as soon as possible (15-20% gain on transfer time)
  • Allow preemption by high priority transfers
Highlight: Execution Time Reduction
Combine Everything with Components

- High Level Component Model (INRIA-Avalon)
  - Hierarchical component model, connector based
  - Generic model (template à la C++)
    - Enable to define MapReduce skeletons!
  - Abstract model (primitive component and connectors)

- A HLCM application is transformed into an executable application
  - Take into account targeted platform (BlobSeer, Bitdew, both, etc.)
  - Insert optimizations (partitioning algorithm are not hard coded in the platform)

- HLCM/L2C: a specialization of HLCM
  - Primitive components in C++ (Java or Python can be added)
  - Primitive connectors: C++ method invocation, CORBA, MPI
    - More primitive connectors are easy to add: WebService, …

- HLCM provides a framework to build an hybrid MapReduce platform
Conclusion

- MapReduce: many issues have not been solved yet!
- Open problems
  - High-throughput data-intensive processing
  - Fault tolerance
  - Optimized scheduling
- Our approach: an integrated architecture for hybrid platforms leveraging efficient bricks: Nimbus, BlobSeer, BitDew
- Future work: integrate and experiment our global architecture on the Grid’5000 testbed, FutureGrid and OpenCirrus