Session 5

Big Data, HPC, and Information Security

Anne Benoit, ENS Lyon, ROMA team

Anne.Benoit@ens-lyon.fr

October 20, 2015, Rennes, France

International Workshop For International Collaboration
On Trustworthy Software

ROMA

Resource Optimization: Models, Algorithms, and Scheduling

Big Data, HPC, Information Security

Team leader: Frédéric Vivien

CNRS · ENS DE LYON · Inria (informatics mathematics) · UCBL Lyon 1

Outline

1. Roma and its research project
2. International collaborations, contracts, etc.
3. Focus on resilience: Which verification for soft error detection?
4. Presentation of the Avalon team

Team composition

Permanent members

CNRS: Loris Marchal (CR) & Bora Uçar (CR)
ENS Lyon: Anne Benoit (MCF, HdR) & Yves Robert (PR, IUF, UTK)
Inria: Jean-Yves L'Excellent (CR, HdR) & Frédéric Vivien (DR, HdR) & Christophe Alias (CR)
Univ. Lyon 1: Laure Gonnord (MCF)

PhD Students

- Aurélien Cavelan
- Julien Herrmann
- Oguz Kaya
- Maroua Maleej
- Loic Pottier
- Bertrand Simon

Administrative assistant

- Laetitia Lecot

Post-Doc and Engineers

- Hongyang Sun
- Chiara Puglisi
- Guillaume Joslin
- Marie Durand

Goal of the Roma team

Aim of the Roma team

- Design models, algorithms, and scheduling strategies to optimize the execution of scientific applications on High-Performance Computing platforms
- Obtain the "best" possible performance from the point of view of the user (e.g., application execution time) while using ressources as efficiently as possible (e.g., low energy consumption)
- Work ranges from theoretical studies to the development of software used daily in the academic and industrial world

Three research themes

1. Application resilience
2. Multi-criteria scheduling strategies
3. Solvers for sparse linear algebra
   and related optimization problems

Application resilience

Applications must be resilient

- Most powerful supercomputers: more than 1 failure per day
- Fault-tolerance techniques: fault prediction, error detection, checkpointing, replication, migration, recovery, etc.
- Resilience: ability to produce correct results in spite of faults

Analysis of fault-tolerance protocols

- Protocols not evaluated through extensive experiments
- Model of platforms, applications, and fault-tolerance protocols
- Question: given an application and a platform, which protocol to use with which parameters?

Algorithm-based fault tolerance (ABFT)

- Focus on direct methods for dense linear algebra kernels
- Extra rows/columns dedicated to fault-tolerance through error-correcting codes
- Trade-off between numerical benefit and cost in resources

Multi-criteria scheduling strategies

Classical approach to application mapping/scheduling

- Minimize absolute performance (e.g., makespan)
- No notion of efficiency nor yield
- May lead to significant waste of resources

Our approach

- Look for a "clever" usage of resources
- Consider multi-criteria optimization
- Trade-offs between
    - User-oriented metrics (QoS)
    - System-oriented metrics (resource usage)

Energy-aware algorithms

- Energy-consumption of fault-tolerance protocols
- Powering cores below nominal voltages + ABFT algorithms

Memory-aware algorithms

- Parallel algorithms to minimize memory-peak usage
- Focus on elimination trees of sparse direct linear solvers
- Graphs of parallel tasks and/or hybrid CPU-GPU platforms

Solvers for sparse linear algebra and related optimization problems (1/2)

Direct solvers for sparse linear systems

- Focus on parallel sparse direct multifrontal methods
- MUMPS software (`http://mumps-solver.org`)

- Addressing massive, hierarchical, parallelism
  - Hybrid parallelism paradigm using both message-passing and multithreading
  - MPI + OpenMP vs. task-based runtime systems such as StarPU or PaRSEC
  - Asynchronism and optimization of communications vs. memory consumption
- Exploitation of low-rank representations
  - Used to compress intermediate dense data structures
  - Study numerical aspects and complexity of factorization and solve
  - Impact of non-predictibility of compression on scheduling

Solvers for sparse linear algebra and related optimization problems (2/2)

Combinatorial scientific computing

Design and analysis of combinatorial algorithms
to enable scientific computing

- Hypergraph partitioning
  - NP-complete problem; existing heuristics have no performance guarantees
  - Design specialized for particular classes of hypergraphs
  - Combine specialized partitioning algorithms with classical multilevel paradigm
- Bipartite matching
  - Maximum cardinality or weighted bipartite matching problem
  - Design parallel heuristics and approximation algorithms
  - Adapt matching algorithms to state-of-the-art computers (multicore, GPU, etc.)

Highlights (presented at the lab evaluation in Nov. 2014)

Awards and visibility

- Y. Robert awarded the *2014 IEEE TCSC Award for Excellence*
- IUF members: A. Benoit (junior, 2009) and Y. Robert (senior, 2011)
- Yves Robert is a member of the "NSF/TCPP Curriculum Initiative on Parallel and Distributed Computing"
- Vice-program chairs for the *Algorithms* tracks of HiPC 2010, HiPC 2012, HiPC 2014, IPDPS'13, IPDPS'14, and SC'14, and for the *Applications* track of ICPP 2011
- Best paper awards at ISPDC'2010 and HeteroPar'2009

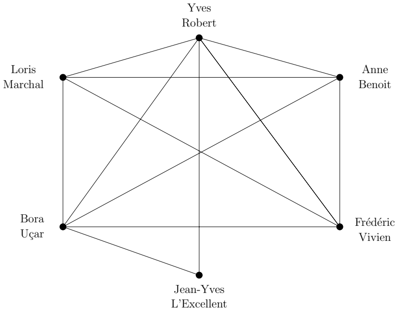
New research themes

- Combinatorial Scientific Computing, following hiring of Bora Uçar as a CNRS CR in January 2009
- Resilience of applications executed on failure-prone platforms

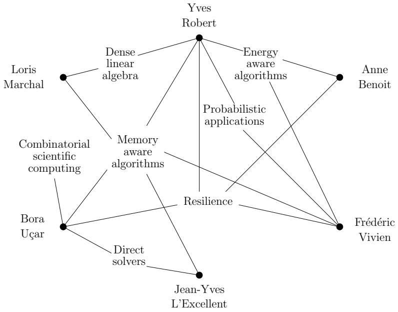
Publications

- Textbook "Fault-Tolerance Techniques for High-Performance Computing", edited by T. Herault and Y. Robert, Springer Verlag, 2015
- Textbook "A Guide to Algorithm Design: Paradigms, Methods, and Complexity Analysis", A. Benoit, Y. Robert, and F. Vivien, Chapman & Hall/CRC, 2013
- Textbook "Introduction to scheduling" edited by Y. Robert and F. Vivien, Chapman & Hall/CRC, 2009
- 48 articles in international peer-reviewed journals
- 89 articles in international peer-reviewed conferences
- 14 book chapters
- 6 special issues of journals, or conference proceedings
- 6 PhD and 2 habilitation theses defended

Relationships between permanent members



Co-publication graph



Relationships between researchers and research themes

Outline

1. Roma and its research project
2. International collaborations, contracts, etc.
3. Focus on resilience: Which verification for soft error detection?
4. Presentation of the Avalon team

Joint Laboratory for Extreme Scale Computing (JLESC)

Partners

- University of Illinois at Urbana-Champaign
- INRIA
- Argonne National Laboratory
- Barcelona Supercomputing Center
- Jülich Supercomputing Centre
- Riken Advanced Institute for Computational Science

Head of JLESC: Franck Cappello (external collaborator of Roma)
Head for INRIA: Yves Robert

Main international collaborations (2009-2014)

- Bilkent University, Turkey: C. Aykanat.
- Ohio State University, USA: Ü. Çatalyürek, K. Kaya, and E. Saule.
- Lawrence Berkeley Laboratory, USA: Xiaoye Sherry Li.
- LSTC, USA: C. Ashcraft.
- University of Hawai'i at Mānoa, USA: H. Casanova.
- Argonne National Laboratory, USA: F. Cappello and M. Snir.
- University of Tennessee, Knoxville, USA: A. Bouteiller, G. Bosilca, J. Dongarra, Th. Hérault, J. Kurzak and P. Luszczek.
- University of Strathclyde, UK: Ph. A. Knight.
- Rutherford Appleton Laboratory, Didcot, UK: I. S. Duff.
- University of Colorado, Denver, USA: J. Langou.
- Washington University in St. Louis, USA: K. Agrawal.
- Northeastern University, USA: A. Rosenberg.
- University of Pittsburgh, USA: R. Melhem.
- University of Auckland, New Zealand: O. Sinnen.

Contracts

- ANR White Project Rescue (2010-2015). Leader: Y. Robert. Project with Grand-Large and Hiepacs. (Application resilience.)
- European FP7 project SCORPIO (2013-2016), 3 years. Project with CERTH, Greece (coordinator); EPFL, Switzerland; RWTH Aachen University, Germany; The Queen's University of Belfast, UK; and IMEC, Belgium. (Application resilience.)
- ANR Project Solhar (2013-2017). Project with HiePACS, Cepage, Runtime, CNRS-IRIT, and two industrial partners: CEA/CESTA and EADS-IW. (Direct solvers.)

Editorial duties (2009-2015)

Editorial committees of journals

- Anne Benoit: *Transactions on Parallel and Distributed Systems (TPDS)*, *Journal of Parallel and Distributed Computing (JPDC)*, and *Journal of Sustainable Computing: Informatics and Systems (SUSCOM)*.
- Yves Robert: *International Journal of High Performance Computing Applications (IJHPCA)*, *International Journal of Grid and Utility Computing (IJGUC)*, and *Journal of Computational Science (JOCS)*.
- Frédéric Vivien: *Parallel Computing*.

4 permanent members of Roma were Vice-program chairs for the *Algorithms* tracks of HiPC'10, HiPC'12, HiPC'14, HiPC'15, IPDPS'13, IPDPS'14, and SC'14, and for the *Applications* track of ICPP'11

Roma permanent members were involved in more than 110 conference PCs (2009-2014)

Teaching

Master level courses at ENS Lyon

- Resilient and energy-aware algorithms: A. Benoit, 2015-2016
- Algorithms for HPC platforms: Frédéric Vivien, 2013-2015.
- Combinatorial scientific computing: Bora Uçar, 2013-2015.
- Parallel algorithms: Anne Benoit, 2007-2010.
- Parallel algorithms and parallel programming: Frédéric Vivien, 2010-2015.
- Scheduling: Loris Marchal, 2008, 2011-2013.
- Sparse matrix computations: Jean-Yves L'Excellent and Bora Uçar, 2009-2011.

License level courses at ENS Lyon

- Algorithms, Advanced algorithms: Anne Benoit and Yves Robert, 2005-2010, 2013-2016.
- Operating systems and networks: Anne Benoit, 2012-2015.
- Probability: Yves Robert, 2010-2013.

Teaching at ECNU

Courses at ECNU by Yves Robert (and Patrice Quinton)

- Parallel algorithms (January 2015)
- Advanced algorithms and complexity (September 2015)

Outline

1 Roma and its research project

2 International collaborations, contracts, etc.

3 Focus on resilience: Which verification for soft error detection?
- Motivation
- Coping with silent errors
- Problem statement
- Theoretical analysis
- Performance evaluation
- Conclusion

4 Presentation of the Avalon team

Outline

1 Roma and its research project

2 International collaborations, contracts, etc.

3 Focus on resilience: Which verification for soft error detection?
- Motivation
- Coping with silent errors
- Problem statement
- Theoretical analysis
- Performance evaluation
- Conclusion

4 Presentation of the Avalon team

Exascale platforms

- Hierarchical
  - 10^5 or 10^6 nodes
  - Each node equipped with 10^4 or 10^3 cores

- Failure-prone

MTBF – one node	1 year	10 years	120 years
MTBF – platform of 10^6 nodes	30sec	5mn	1h

More nodes ⇒ Shorter MTBF (Mean Time Between Failures)

- Energy efficiency

  Thermal power close to the one of a nuclear reactor!
  A critical issue to address if we want to achieve Exascale.

Exascale platforms

- Hierarchical
  - 10^5 or 10^6 nodes
  - Each node equipped with 10^4 or 10^3 cores

- Failure-prone

MTBF – one node	1 y[illegible]	10 years	120 years
MTBF – platform of 10^6 nodes	[illegible]	5mn	1h

More nodes ⇒ Shorter MTBF (Mean Time Between Failures)

- Energy efficiency

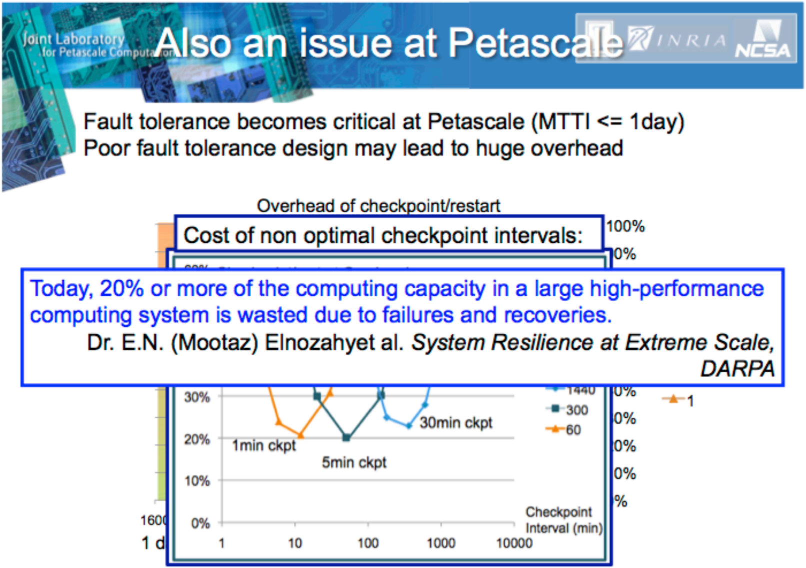
  Thermal po[illegible] [illegible]tor!

  A critical is[illegible] [illegible]e Exascale.

Exascale

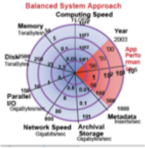
$\neq$ Petascale $\times 1000$

Even for today's platforms (courtesy F. Cappello)

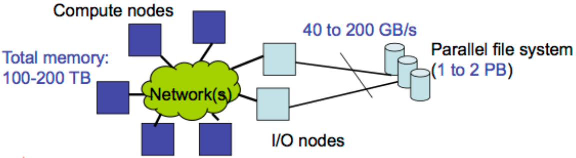


Even for today’s platforms (courtesy F. Cappello)

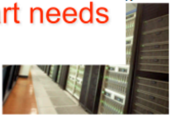
Classic approach for FT: Checkpoint-Restart



Typical “Balanced Architecture” for PetaScale Computers



RoadRunner



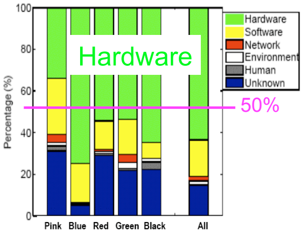
Without optimization, Checkpoint-Restart needs about 1h! (~30 minutes each)

Systems	Perf.	Ckpt time	Source
RoadRunner	1PF	~20 min.	Panasas
LLNL BG/L	500 TF	>20 min.	LLNL
LLNL Zeus	11TF	26 min.	LLNL
YYY BG/P	100 TF	~30 min.	YYY

LLNL BG/L

Sources of failures

- Analysis of error and failure logs
- In 2005 (Ph. D. of CHARNG-DA LU) : "Software halts account for the most number of outages (59-84 percent), and take the shortest time to repair (0.6-1.5 hours). Hardware problems, albeit rarer, need 6.3-100.7 hours to solve."
- In 2007 (Garth Gibson, ICPP Keynote):



Relative frequency of root cause by system type.

- In 2008 (Oliner and J. Stearley, DSN Conf.):

Type	Raw Count	Raw %	Filtered Count	Filtered %
Hardware	174,586,516	98.04	1,999	18.78
Software	144,899	0.08	6,814	64.01
Indeterminate	3,350,044	1.88	1,832	17.21

Software errors: Applications, OS bug (kernel panic), communication libs, File system error and other.

Hardware errors, Disks, processors, memory, network

Conclusion: Both Hardware and Software failures have to be considered

A few definitions

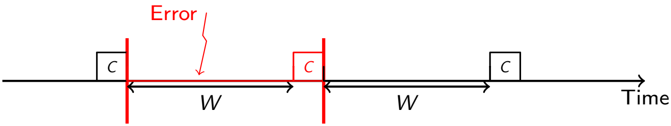
- Many types of failures: software error, hardware malfunction, memory corruption
- Many possible behaviors: silent, transient, unrecoverable
- Restrict to failures that lead to application failures
- This includes all hardware failures, and some software ones

Outline

1. Roma and its research project
2. International collaborations, contracts, etc.
3. Focus on resilience: Which verification for soft error detection?
   - Motivation
   - Coping with silent errors
   - Problem statement
   - Theoretical analysis
   - Performance evaluation
   - Conclusion
4. Presentation of the Avalon team

General-purpose approach

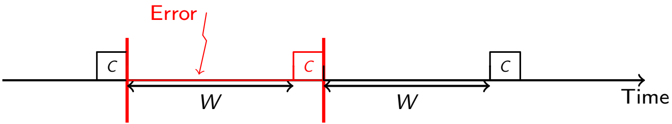
Periodic checkpoint, rollback and recovery:



- Fail-stop errors: instantaneous error detection, e.g., resource crash

General-purpose approach

Periodic checkpoint, rollback and recovery:



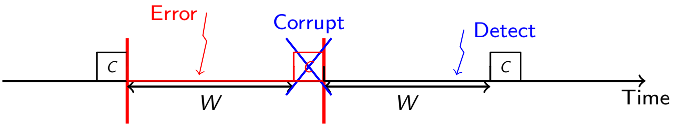
- Fail-stop errors: instantaneous error detection, e.g., resource crash
- Silent errors (aka silent data corruptions): e.g., soft faults in L1 cache, ALU, double bit flip

Silent error is detected only when corrupted data is activated, which could happen long after its occurrence

Detection latency is problematic $\Rightarrow$ risk of saving corrupted checkpoint!

General-purpose approach

Periodic checkpoint, rollback and recovery:



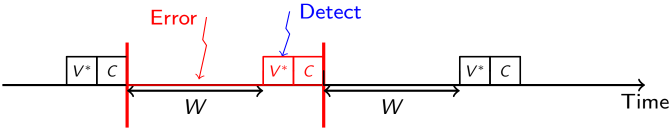
- Fail-stop errors: instantaneous error detection, e.g., resource crash
- Silent errors (aka silent data corruptions): e.g., soft faults in L1 cache, ALU, double bit flip

Silent error is detected only when corrupted data is activated, which could happen long after its occurrence

Detection latency is problematic $\Rightarrow$ risk of saving corrupted checkpoint!

Coping with silent errors

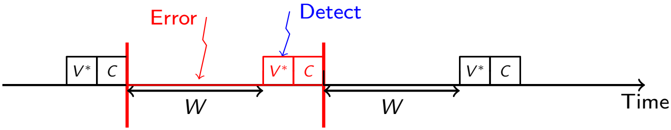
Couple checkpointing with verification:



- Before each checkpoint, run some verification mechanism (checksum, ECC, coherence tests, TMR, etc)
- Silent error is detected by verification ⇒ checkpoint always valid ☺

Coping with silent errors

Couple checkpointing with verification:



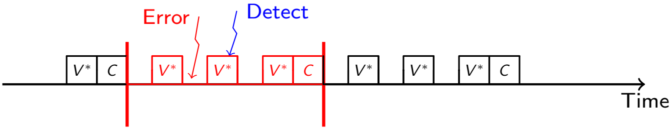
- Before each checkpoint, run some verification mechanism (checksum, ECC, coherence tests, TMR, etc)
- Silent error is detected by verification ⇒ checkpoint always valid ☺

Optimal period (Young/Daly):

	Fail-stop (classical)	Silent errors
Pattern	$T = W + C$	$T = W + V^* + C$
Optimal	$W^* = \sqrt{2C\mu}$	$W^* = \sqrt{(C + V^*)\mu}$

One step further

Perform several verifications before each checkpoint:



- Pro: silent error is detected earlier in the pattern ☺
- Con: additional overhead in error-free executions ☹

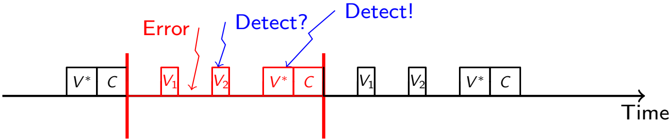
How many intermediate verifications to use and the positions?

Partial verification

Guaranteed/perfect verifications (V^*) can be very expensive!

Partial verifications (V) are available for many HPC applications!

- Lower accuracy: recall $r = \frac{\#\text{detected errors}}{\#\text{total errors}} < 1$ ☹
- Much lower cost, i.e., $V < V^*$ ☺



Which verification(s) to use? How many? Positions?

Outline

1. Roma and its research project
2. International collaborations, contracts, etc.
3. Focus on resilience: Which verification for soft error detection?
   - Motivation
   - Coping with silent errors
   - Problem statement
   - Theoretical analysis
   - Performance evaluation
   - Conclusion
4. Presentation of the Avalon team

Model and objective

Silent errors

- Poisson process: arrival rate $\lambda = 1/\mu$, where μ is platform MTBF
- Strike only computations; checkpointing, recovery, and verifications are protected

Resilience parameters

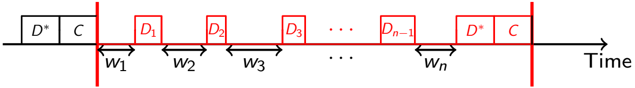
- Cost of checkpointing C, cost of recovery R
- k types of partial detectors and a perfect detector $\left(D^{(1)}, D^{(2)}, \ldots, D^{(k)}, D^*\right)$
  - $D^{(i)}$: cost $V^{(i)}$ and recall $r^{(i)} < 1$
  - D^*: cost V^* and recall $r^* = 1$

Design an optimal periodic computing pattern that minimizes execution time (or makespan) of the application

Pattern

Formally, a pattern $\textsc{Pattern}(W, n, \boldsymbol{\alpha}, \mathbf{D})$ is defined by

- W: pattern work length (or period)
- n: number of work segments, of lengths w_i (with $\sum_{i=1}^{n} w_i = W$)
- $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n]$: work fraction of each segment ($\alpha_i = w_i/W$ and $\sum_{i=1}^{n} \alpha_i = 1$)
- $\mathbf{D} = [D_1, D_2, \ldots, D_{n-1}, D^*]$: detectors used at the end of each segment ($D_i = D^{(j)}$ for some type j)



- Last detector is perfect to avoid saving corrupted checkpoints

- The same detector type $D^{(j)}$ could be used at the end of several segments

Outline

1. Roma and its research project
2. International collaborations, contracts, etc.
3. Focus on resilience: Which verification for soft error detection?
   - Motivation
   - Coping with silent errors
   - Problem statement
   - Theoretical analysis
   - Performance evaluation
   - Conclusion
4. Presentation of the Avalon team

Summary of results

In a nutshell:

- Given a pattern $\text{PATTERN}(W, n, \boldsymbol{\alpha}, \mathbf{D})$,
  - We show how to compute the expected execution time
  - We are able to characterize its optimal length
  - We can compute the optimal positions of the partial verifications

Summary of results

In a nutshell:

- Given a pattern $\textsc{Pattern}(W, n, \boldsymbol{\alpha}, \mathbf{D})$,
  - We show how to compute the expected execution time
  - We are able to characterize its optimal length
  - We can compute the optimal positions of the partial verifications
- However, we prove that finding the optimal pattern is NP-hard
- We design an FPTAS (Fully Polynomial-Time Approximation Scheme) that gives a makespan within $(1 + \epsilon)$ times the optimal with running time polynomial in the input size and $1/\epsilon$
- We show a simple greedy algorithm that works well in practice

Summary of results

Algorithm to determine a pattern $\textsc{Pattern}(W, n, \boldsymbol{\alpha}, \mathbf{D})$:

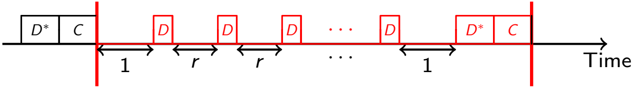
- Use FPTAS or Greedy (or even brute force for small instances) to find (optimal) number n of segments and set $\mathbf{D}$ of used detectors
- Arrange the $n-1$ partial detectors in any order
- Compute $W^* = \sqrt{\frac{o_{\text{ff}}}{\lambda f_{\text{re}}}}$ and $\alpha_i^* = \frac{1}{U_n} \cdot \frac{1-g_{i-1}g_i}{(1+g_{i-1})(1+g_i)}$ for $1 \leq i \leq n$,

$$\text{where } o_{\text{ff}} = \sum_{i=1}^{n-1} V_i + V^* + C \text{ and } f_{\text{re}} = \frac{1}{2}\left(1 + \frac{1}{U_n}\right)$$

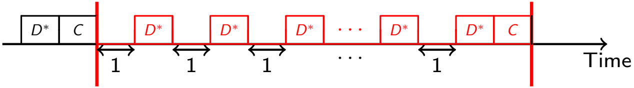
$$\text{with } g_i = 1 - r_i \text{ and } U_n = 1 + \sum_{i=1}^{n-1} \frac{1-g_i}{1+g_i}$$

Two special cases

- When all verifications use the same partial detector (r), we get
$$\alpha_k^* = \begin{cases} \frac{1}{(n-2)r+2} & \text{for } k = 1 \text{ and } k = n \\ \frac{r}{(n-2)r+2} & \text{for } 2 \le k \le n-1 \end{cases}$$



- When all verifications use the perfect detector, we get equal-length segments, i.e., $\alpha_k^* = \frac{1}{n}$ for all $1 \le k \le n$



Optimal number and set of detectors

It remains to determine optimal n and $\mathbf{D}$ of a pattern $\textsc{Pattern}(W, n, \boldsymbol{\alpha}, \mathbf{D})$.

Optimal number and set of detectors

It remains to determine optimal n and $\mathbf{D}$ of a pattern $\textsc{Pattern}(W, n, \boldsymbol{\alpha}, \mathbf{D})$.

Equivalent to the following optimization problem (determine the m_j's, or equivalently, a vector $\mathbf{m}$):

$$\text{Minimize} \qquad f_{\text{re}} o_{\text{ff}} = \frac{V^* + C}{2} \left(1 + \frac{1}{1 + \sum_{j=1}^{k} m_j a^{(j)}}\right) \left(1 + \sum_{j=1}^{k} m_j b^{(j)}\right)$$

$$\text{subject to} \qquad m_j \in \mathbb{N}_0 \quad \forall j = 1, 2, \dots, k$$

accuracy: $a^{(j)} = \dfrac{1 - g^{(j)}}{1 + g^{(j)}}$ relative cost: $b^{(j)} = \dfrac{V^{(j)}}{V^* + C}$

accuracy-to-cost ratio: $\phi^{(j)} = \dfrac{a^{(j)}}{b^{(j)}}$

NP-hard even when all detectors share the same accuracy-to-cost ratio (reduction from unbounded subset sum), but admits an FPTAS.

Greedy algorithm

Practically, a greedy algorithm:

- Employs only the detector with highest accuracy-to-cost ratio $\phi^{\max} = \frac{a}{b}$

$$\text{Optimal number of detectors: } m^* = -\frac{1}{a} + \sqrt{\frac{1}{a}\left(\frac{1}{b} - \frac{1}{a}\right)}$$

$$\text{Optimal overhead: } H^* = \sqrt{\frac{2(C + V^*)}{\mu}}\left(\sqrt{\frac{1}{\phi^{\max}}} + \sqrt{1 - \frac{1}{\phi^{\max}}}\right)$$

- Rounds up the optimal rational solution $\lceil m^* \rceil$

The greedy algorithm has an approximation ratio $\sqrt{3/2} < 1.23$

Outline

1 Roma and its research project

2 International collaborations, contracts, etc.

3 Focus on resilience: Which verification for soft error detection?
- Motivation
- Coping with silent errors
- Problem statement
- Theoretical analysis
- Performance evaluation
- Conclusion

4 Presentation of the Avalon team

Simulation configuration

Exascale platform:

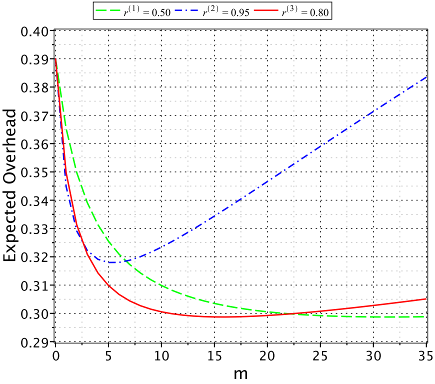
- 10^5 computing nodes with individual MTBF of 100 years
  $\Rightarrow$ platform MTBF $\mu \approx 8.7$ hours
- Checkpoint sizes of 300GB with throughput of 0.5GB/s
  $\Rightarrow C = 600s$

Realistic detectors (designed at ANL):

	cost	recall	ACR
Time series prediction $D^{(1)}$	$V^{(1)} = 3s$	$r^{(1)} = 0.5$	$\phi^{(1)} = 133$
Spatial interpolation $D^{(2)}$	$V^{(2)} = 30s$	$r^{(2)} = 0.95$	$\phi^{(2)} = 36$
Combination of the two $D^{(3)}$	$V^{(3)} = 6s$	$r^{(3)} = 0.8$	$\phi^{(3)} = 133$
Perfect detector D^*	$V^* = 600s$	$r^* = 1$	$\phi^* = 2$

Evaluation results

Using individual detector (greedy algorithm)



Best partial detectors offer ∼9% improvement in overhead.
Saving ∼55 minutes for every 10 hours of computation!

Evaluation results

Mixing two detectors: depending on application or dataset, a detector's recall may vary, but its cost stays the same

Realistic data again!
$r^{(1)} = [0.5, 0.9]$
$r^{(2)} = [0.75, 0.95]$
$r^{(3)} = [0.8, 0.99]$

$\phi^{(1)} = [133, 327]$
$\phi^{(2)} = [24, 36]$
$\phi^{(3)} = [133, 196]$

	m	overhead H	diff. from opt.
Scenario 1: $r^{(1)} = 0.51$, $r^{(3)} = 0.82$, $\phi^{(1)} \approx 137$, $\phi^{(3)} \approx 139$			
Optimal solution	(1, 15)	29.828%	0%
Greedy with $D^{(3)}$	(0, 16)	29.829%	0.001%
Scenario 2: $r^{(1)} = 0.58$, $r^{(3)} = 0.9$, $\phi^{(1)} \approx 163$, $\phi^{(3)} \approx 164$			
Optimal solution	(1, 14)	29.659%	0%
Greedy with $D^{(3)}$	(0, 15)	29.661%	0.002%
Scenario 3: $r^{(1)} = 0.64$, $r^{(3)} = 0.97$, $\phi^{(1)} \approx 188$, $\phi^{(3)} \approx 188$			
Optimal solution	(1, 13)	29.523%	0%
Greedy with $D^{(1)}$	(27, 0)	29.524%	0.001%
Greedy with $D^{(3)}$	(0, 14)	29.525%	0.002%

The greedy algorithm works very well in this practical scenario!

Outline

1 Roma and its research project

2 International collaborations, contracts, etc.

3 Focus on resilience: Which verification for soft error detection?
- Motivation
- Coping with silent errors
- Problem statement
- Theoretical analysis
- Performance evaluation
- Conclusion

4 Presentation of the Avalon team

Conclusion

A first comprehensive analysis of computing patterns with partial verifications to detect silent errors

- Theoretically: assess the complexity of the problem and propose efficient approximation schemes
- Practically: present a greedy algorithm and demonstrate its good performance with realistic detectors

Future directions

- Partial detectors with false positives/alarms

$$\text{precision } p = \frac{\#\text{true errors}}{\#\text{detected errors}} < 1$$

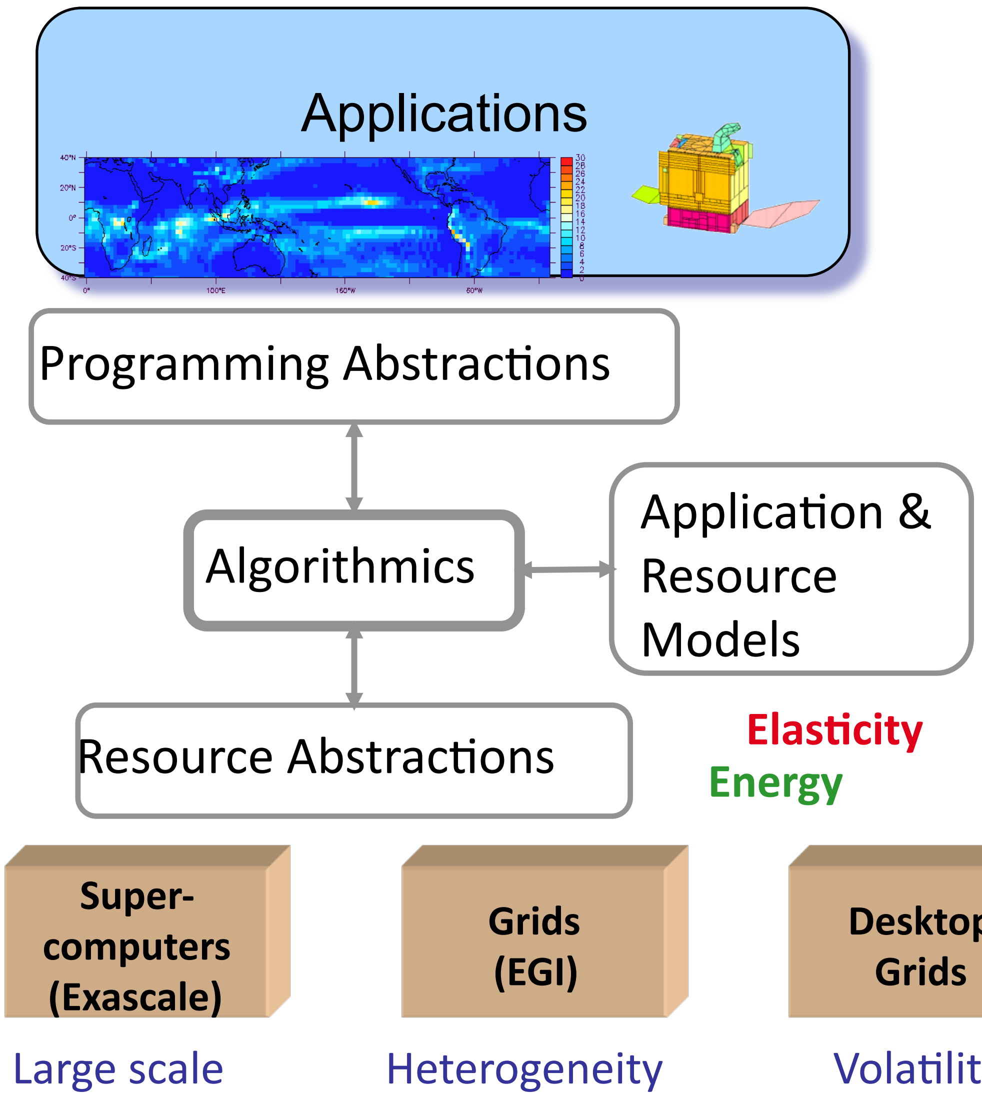
- Errors in checkpointing, recovery, and verifications
- Coexistence of fail-stop and silent errors

Research report available at https://hal.inria.fr/hal-01164445v1

Outline

1. Roma and its research project
2. International collaborations, contracts, etc.
3. Focus on resilience: Which verification for soft error detection?
4. Presentation of the Avalon team

Avalon: Research Activities



CPU/data-intensive Scientific Applications

- From “simple” to code coupling
  - Structure complexity
  - “New” forms of interactions (MR)

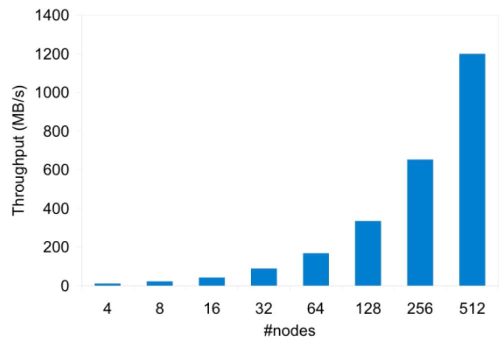
Computing platforms

- Different characteristics
  - Performance, energy, size, cost, reliability, QoS, etc.
- Hybridization
  - Sky computing, HPC@Cloud, Exascale, Spot instance

Objectives

- Expressiveness simplicity
- Application portability
- Resource specific optimizations
  - Elastic resource management
  - Energy consumption

MapReduce for Large, Distributed, and Dynamic Datasets



Throughput of WordCount application on Grid'5000 (512 nodes) up to 2 TB

MapReduce runtime for

- Distributed over hybrid and widely distributed infrastructures
  - Cloud, Desktop PCs, sensors, smartphones…
- Dynamic, i.e. that grow or shrink during time, or partially unavailable because of infrastructure failures.

MapReduce, Beyond the Data Center BitDew/Active Data

- First implementation of MapReduce for Internet Desktop Grid
  - 2-level scheduler, latency hiding, p-failures resilient, collective communications
- Algorithm distributed result checking of intermediate
- MapReduce/ActiveData: incremental processing of dynamic datasets
- Storage on hybrid Cloud + Desktop PCs nodes
- Privacy computing on hybrid infrastructures using Information Dispersal Algorithms
- MapReduce for Hybrid Infrastructures : Desktop Grids + Clouds
  - BigHybrid : simulator based on SimGrid
  - Software prototype : MapReduce/BitDew + Hadoop/Blobseer
- Network distance aware data placement

> 15 publications including: FGCS'15, CCPE'15, CCPE'15, ICA3PP'15, PDP'15,DataCom'15, ICA3PP'14, GLOBE'14, ….

SFSysLab: Sino-French Joint Research Center on Systems for Large Scale Computing and Data Management

- Université Paris Sorbonne Cité, Paris (C.Cérin)
- INRIA/Ecole Normale Supérieure, Lyon (G. Fedak)
- INRIA/IRISA, Rennes (S. Ibrahim)
- Chinese Academy of Science/CNIC, Beijing (H. He)
- Huazong University of Science and Technology , Wuhan(X. Shi)
- Hangzhou Dianzi University, (C. Jiang)

Research Topics

Theme 1: **Middleware for data management**
Data management; Data life cycle;
Data-aware toolkits and middleware;
Scheduling and management; Formal modeling;

Theme 2**: HPC and Data Science**
Parallel processing techniques for big data analysis;
Clusters, Grids and Cloud computing for big data processing;
High performance data transfer and ingestion

Theme 3: **Machine Learning, Storage and Systems for data management**
GPU algorithms for deep learning;
AI systems for handling big data;

Theme 4**: Mobile computing and data management**
Networking support; Data and information;
Energy-aware data management

Theme 5: **Applications**
Data-intensive applications;
Preservation; Stream Data processing;

MapReduce Master Class
Design, Performance, Optimizations

Gilles Fedak

This course covers the *MapReduce programming model* and its eco-systems as well as the challenges of designing efficient Big Data middleware and applications : Big Data concepts, technologies (Hadoop, HDFS, Hbase, Pig, Spark), research challenges around MapReduce, large-scale Big Data.

- **University Babeș-Bolya**, CLuj Napoca, Romania, 4-6 Novembre 2014
  - 9 hours including Big Data related topics
- **Université Paris XIII** - Formation doctorale de l'institut Galilé, 1 Avril 2014
  - 8 hours including practice
- **Ecole Normale Supérieure de Lyon** - Master Informatique, 2013, 2014, 2015
- Il Escola Regional de Alto Desempenho - Região Nordeste, **Savaldor de Bahia, Brazil**, October 22, 2013
- Seminar Datenverarbeitung mit Map-Reduce, **Univ. of Heidelberg, Germany**, 2012.

2015-16 (planned)

- University of Paris Sorbonne Cité
- Chinese Academy of Science (Beijing, CAS President International Fellowship Initiative PIFI)