第１１回先進スーパーコンピューティング環境研究会（ASE 研究会）実施報告

東京大学情報基盤センター　准教授　片桐孝洋

2012 年 3 月 16 日（金）13 時 00 分～17 時 25 分、東京大学情報基盤センター４階遠隔講義室にて、第 11 回先進スーパーコンピューティング環境研究会（ASE 研究会）が開催されました。

国内の大学・研究機関、および企業からの参加者は 24 名でした。活発な議論がなされました。

招待講演として、University of Lille, CNRS / LIL and INRIA の Serge G. Petiton 教授をお呼びし、クリロフ部分空間法の自動チューニングに関する講演を行いました。

Petiton 教授の講演は、「Toward future smart auto-tuned Parallel Krylov methods and programming for Exascale Computing」と題し、近年、数値計算アルゴリズムやライブラリの研究開発で注目されている自動チューニング（Auto-tuning, AT）技術の発表でした。疎行列反復解法のうち、最も強力、かつ有効な方法の１つである GMRES(m)法に対し、新しい AT 方式を適用した研究の紹介がなされました。今後の AT 研究の方向性として、さまざまな AT に必要となるパラメタを、数値解析上や計算機ハードウェア上の情報から、適切に、かつ実行時に調整する Smart Tuning という AT フレームワークを提案しています。この方法では、今後来たる計算資源が有り余るような状況において対象を複数同時実行する場合、よさそうなパラメタを動的に交換しあい、実行ごとに高速化・安定化されていく AT フレームワークが示されました。大変興味深い AT 手法として、聴講いたしました。

東京大学情報基盤センターが実施する「若手利用者推薦制度」の H23 年度前期課題採択者による成果発表も行われました。新進気鋭の若手研究者による、興味深い成果が多数発表されました。また、質疑も活発になされました。

招待講演の内容について、本原稿の後に当日の発表資料を掲載します。ご興味のある方はご覧ください。また、当日のプログラムを以下に載せます。

Program

- 13:00～14:00 Invited Speaker:

  Professor Serge G. Petiton (University of Lille, CNRS / LIL and INRIA)

  Title: Toward future smart auto-tuned Parallel Krylov methods and programming for Exascale Computing

  Abstract: Auto-tuning Krylov subspaces sizes at runtime for the GMRES(m) methods may be efficient to minimize the global computing time toward convergence. We introduce a general algorithm to auto-tuned the Krylov subspace sizes at run time, and we also introduce the first algorithm to auto-tune at run-time the number of vectors targeted by an incomplete orthogonalizations of Krylov basis associated with

the GMRES(m) method, minimizing in particular the number of dot-products for a fixed subspace size. We present and analyse in this talk these new algorithms and we propose some experimental results. As a conclusion, we introduce some future projected researches which would lead to smart auto-tuned Hybrid Krylov methods for exascale computing, including language issues.

■平成23年度前期　若手利用者推薦　成果報告会

- 14:05～14:25 Junichiro Shiomi (Department of Mechanical Engineering, The University of Tokyo)

Title: Calculations of phonon transport in semiconductors based on first principles

Abstract: Encouraged by the recent progress in nanostructuring techniques to enhance the conversion efficiency of thermoelectric materials by reduction of lattice thermal conductivity, we aim to develop a numerical tool that evaluates microscopic heat conduction characteristics, namely phonon transport properties, based on first-principles. The capabilities and limits of the new framework for designing thermoelectrics will be discussed.

- 14:30～14:50 Kimio Kuramitsu (Yokohama National University)

Title: A scripting language design and implementation for HPC computing

Abstract: This talk presents our attempt to extending Konoha scripting language for HPC computing. Konoha is a newly designed scripting language that has static typing feature. As a static typing helps produce well-optimized execution code, Konoha is expected to run scripts much faster than existing dynamic scripting languages, such as Python and Ruby. The speaker will report our experimental results with several efforts, including GPGPU integration, LLVM-base JIT compilation, and MPI-based extension. The myth of HPC scripting will be revealed.

- 14:55～15:15 宮下秀樹（山梨大学大学院　医学工学総合教育部）

タイトル：クラスタシステムにおける高速な特異値分解のための前処理、後処理の実装

概要：特異値分解はさまざまな分野で利用される重要な数値計算アルゴリズムであり、コンピュータの性能向上に伴ないますます大規模化の要求が高くなっている。最新の特異値分解アルゴリズムは主として二重対角行列向けなので、一般行列の特異値分解には前処理として二重対角化が必要となる。この前処理には QR 分解が使用される。
本研究の目的は T2K オープンスパコン上で、OpenMP と MPI による Hybrid プログラミングにより QR 分解を実装することである。

- 15:20～15:40 Hidetaka Muguruma and Yuichi Tsujita (Department of Electronic

Engineering and Computer Science, Faculty of Engineering, Kinki University)

Title: Study of A Pipelined Processing for High Throughput Collective MPI-IO

Abstract: One of the MPI-IO implementations named ROMIO provides high performance collective MPI-IO with the help of its Two-Phase I/O protocol. Two-Phase I/O consists of repetitive operations including file accesses and data communications. We propose a pipelined processing implementation by overlapping file accesses with communications for further throughput improvements. We have evaluated its performance on the T2K Open Supercomputer with a Lustre file system. We observed effectiveness of our implementation. Furthermore, we found effective memory utilization in some access patterns compared with the original Two-Phase I/O.

■Regular Presentations

- 15:45～16:30 Yoshikazu Kamoshida (Information Technology Center, The University of Tokyo)

  Title::Jitter Quantification on a Supercomputer in Operation

  Abstract: Analyzing and preventing system noises on the parallel computing environment are one of very important problems to run the parallel applications with better scalability. We introduce an analysis method of system noises which estimates the processes that are causing the delays of the parallel application and the degree of them. We also discuss about our light-weight monitoring technique which enables our method to work efficiently on the running supercomputer.

- 16:35-17:20 Takahiro Katagiri (Information Technology Center, The University of Tokyo)

  Title: New Auto-tuning Functions and Its Effect of Xabclib and OpenATLib Ver.1.0

Abstract: We have developed a sparse iterative solver with an auto-tuning (AT) facility, named Xabclib. In this presentation, we present new functions of the AT ver. 1.0. The main functions in Xabclib and OpenATLib Ver.1.0 are summarized as follows: First, we have developed new AT for selection of preconditioners and solver algorithms. It is known that selection of preconditioners is crucial function for iterative solvers; however, automatic selection was very difficult in general. We propose an algorithm for detecting stagnation of series of residual errors. By utilizing the algorithm of stagnation detecting, the solver establishes high convergence ratio to that of conventional solvers. Second, but this is not new, we have developed an automatic selection of implementations for sparse matrix vector multiplications (SpMV). Especially, we provide a new implementation of segmented scan (SS) method for scalar processors. This enables us a dynamic parameter change

for the new SS on SpMV. We show several results of performance evaluation with the ver.1.0 with one node (16 threads) of the T2K Open supercomputer (U.Tokyo).

- 17: 25 Closing: Takahiro Katagiri (The University of Tokyo)
- 18:00～ A Banquet near Nedu station

図１　当日の会場の様子

ASE研究会の開催情報はメーリングリストで発信をしております。研究会メーリングリストに参加ご希望の方は、ASE研究会幹事の片桐（katagiri@cc.u-tokyo.ac.jp）までお知らせください。

以上

Toward Future Smart Auto-tuned Parallel Krylov Methods and Programming for Exascale Computing

Serge G. Petiton (LIFL/CNRS)
serge.petiton@lifl.fr
na.spetiton@na.ornl.gov

Outline

- Introduction
- Exascale challenges
- Krylov methods and auto-tuning algorithms
- Incomplete orthogonalization auto-tuning algorithms
- Experimental results
- Hybrid asynchronous krylov methods
- Towards smart-tuning and future intelligent numerical methods
- Conclusion

Outline

- Introduction
- **Exascale challenges**
- Krylov methods and auto-tuning algorithms
- Incomplete orthogonalization auto-tuning algorithms
- Experimental results
- Hybrid asynchronous krylov methods
- Towards smart-tuning and future intelligent numerical methods
- Conclusion

The Future Exaflop barrier : not only another symbolic frontier coming after the Petaflops

- Sustained Petascale applications on an unique computer exist since a few months, the “K” computer reached more than 10 Petaflops on LINPACK,
- Gordon Bell award : 3 sustained petafops (Boku et al.)
- Next frontier : **Exascale** computing (and how many MWatts???)
- Nevertheless, many challenges would emerge, probably before the announced 100 Petaflop computers and beyond.
- We have to be able to **anticipate solutions** to be able to educate scientists as soon as possible to the future programming.
- We have to use the existing emerging platforms and prototype to imagine the future language, systems, algorithms,….
- We have to propose **new programming paradigms** (SPMD/MPI for 1 million of cores and 1 billions of threads????),
- We have to propose **new languages**.
- Co-design and **domain application languages and/or high level multi-level language** and frameworks,…….

New methods for future supercomputer

- We have to imagine new methods for the exascale computers
- Methods would define new architectures (co-design), not the old and present methods….
- Many people propose new system and language starting from the exiiting methods and numerical libraries …. but they were developed for MPI-like programming and only SPMD paradigm, and at the “old time” of the Moore law.
- We have to adapt the methods with respect to criteria from the architecture, the arithmetic, the system, the I/O, the latence,…… then, auto-tuning is becoming a general approach.
- We have to hybridize numerical methods to solve large scientific applications, asynchronously, and each of then have to be auto-tuned,
- We have to find smarter criteria, some of them at the application and at the mathematical level, for each method : **smart tuning**
- These auto-tuned methods will be correlated : **intelligent numerical methods.**

Outline

- Introduction
- Exascale challenges
- **Krylov methods and auto-tuning algorithms**
- Incomplete orthogonalization auto-tuning algorithms
- Experimental results
- Hybrid asynchronous krylov methods
- Towards smart-tuning and future intelligent numerical methods
- Conclusion

GMRES : about memory space and dot products

1. *Start*: Choose x_0 and compute $r_0 = f - Ax_0$ and $v_1 = r_0/\|r_0\|$.
2. *Iterate*: For $j = 1, 2, \cdots, m$ do:
   $h_{i,j} = (Av_j, v_i), i = 1, 2, \cdots, j,$
   $\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j} v_i,$
   $h_{j+1,j} = \|\hat{v}_{j+1}\|$, and
   $v_{j+1} = \hat{v}_{j+1}/h_{j+1,j}$.
3. *Form the approximate solution*:
   $x_m = x_0 + V_m y_m$, where y_m minimizes $\|\beta e_1 - \bar{H}_m y\|$, $y \in R^m$.
4. *Restart*:
   Compute $r_m = f - Ax_m$; if satisfied then stop
   else compute $x_0 := x_m$, $v_1 := r_m/\|r_m\|$ and go to 2.

Memory space :
- sparse matrix : nnz (i.e. < C n) elements
- Krylov basis vectors : n m
- Hessenberg matrix : m m

Scalar products, at j fixed:
- Sparse Matrix-vector product : n of size C
- Orthogonalization : m of size n

m, the subspace size, may be auto-tuned at runtime to minimize the space memory occupation and the number of scalar product, with better or approximately same convergence behaviors, with a minimized computing time.

GMRES : about memory space and dot products

1. *Start*: Choose x_0 and compute $r_0 = f - Ax_0$ and $v_1 = r_0/\|r_0\|$.
2. *Iterate*: For $j = 1, 2, \cdots, m$ do:
   $h_{i,j} = (Av_j, v_i), i = 1, 2, \cdots, j,$
   $\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j} v_i,$
   $h_{j+1,j} = \|\hat{v}_{j+1}\|$, and
   $v_{j+1} = \hat{v}_{j+1}/h_{j+1,j}$.
   Incomplete orthogonalization : sum i =j-q,j ; q>0.
   Then, q+1 bands on the Hesseberg matrix
3. *Form the approximate solution*:
   $x_m = x_0 + V_m y_m$, where y_m minimizes $\|\beta e_1 - \bar{H}_m y\|$, $y \in R^m$.
4. *Restart*:
   Compute $r_m = f - Ax_m$; if satisfied then stop
   else compute $x_0 := x_m$, $v_1 := r_m/\|r_m\|$ and go to 2.

Memory space :
- sparse matrix : nnz (i.e. < C n) elements
- Krylov basis vectors : n m
- Hessenberg matrix : m m

Scalar products, at j fixed:
- Sparse Matrix-vector product : n of size C
- Orthogonalization : m of size n

m, the subspace size, may be auto-tuned at runtime to minimize the space memory occupation and the number of scalar product, with better or approximately same convergence behaviors. The number of vectors othogonalized with the new one may be auto-tuned at runtime.

Auto-tuned Krylov methods, some correlated goals

- Minimize the global computing time,
- Accelerate the convergence,
- Minimize the number of communications,
- Minimize the number of longer size scalar products,
- Minimize memory space,
- Select the best sparse matrix compressed format,
- Mixed arithmetic.

Criteria which are some of the requirements for Petascale and future Exascale computing.

The goal of this talk is to illustrate the difficulties to analyze auto-tuning algorithm efficiency and to conclude that we would need "smart" auto-tuning algorithms to create the next generation of High Performance Numerical software.

Experiences on cluster of GPU confirm the difficulties to conclude with the today auto-tuning algorithms, to many correlated criteria have to be analyzed and it is quite impossible to have some stable "conclusions"

Previous works

- Subspace size : different auto-tuning at runtime
  - Subspace size increase, until a fixed limit [Katagiri00][Sosonkina96]
  - Subspace size decrease, until a fixed limit [Baker09]
  - Restart Trigger [Zhang04], restart when stagnation is detected.
- Orthogonalization : no auto-tuning at runtime
  - Prior to execution : [Jia94]

Remark, in general:

- Greater subspace size -> better convergence/long restart, less iterations
- Smaller subspace size -> slow convergence, stagnation, short restart, more iteration
- Choice of m is mandatory.

Subspace size tuning principle

We both increase and decrease the subspace size: based on the adaptive subspace size

$$Cr = norm2\ (r_i) / norm2(r_{i\_1})$$

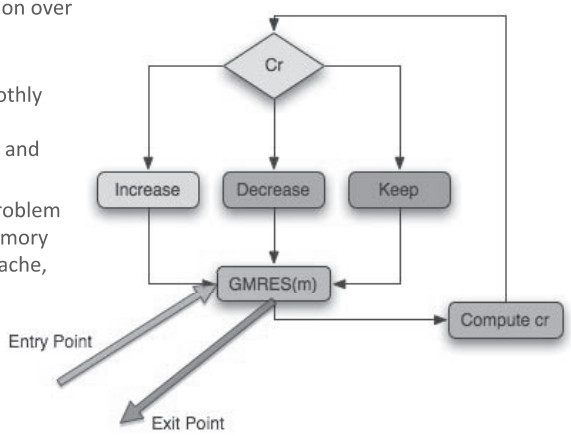
1 - Keep previous subspace size (cr low) :satisfying

2 - Decrease subspace size (cr medium) : reduce cycle time, we will have approximately the same convergence rate but with less operations and communications per restart.

3 - Increase restart (cr high) : we want to accelerate the convergence
• Track memory levels : Cache, RAM, Nodes

Auto-Tuning Algorithms

- Subspace size
  - Evaluate convergence progression over m iterations.
  - Decrease if convergence are monotonous or if they are smoothly slowing (approximately same convergence but minimize time and space)- **Cr medium**
  - Increase if convergence stall (problem if we increase too much the memory space), Track memory levels : Cache, RAM, Nodes. **Cr low**
  - Do nothing if **Cr high**

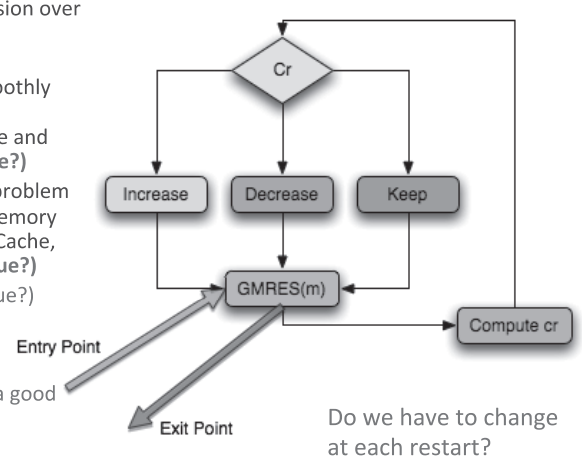


$Cr = norm2\ (r_i) / norm2(r_{i\_1})$

Easy to implement using libraries

Auto-Tuning Algorithms

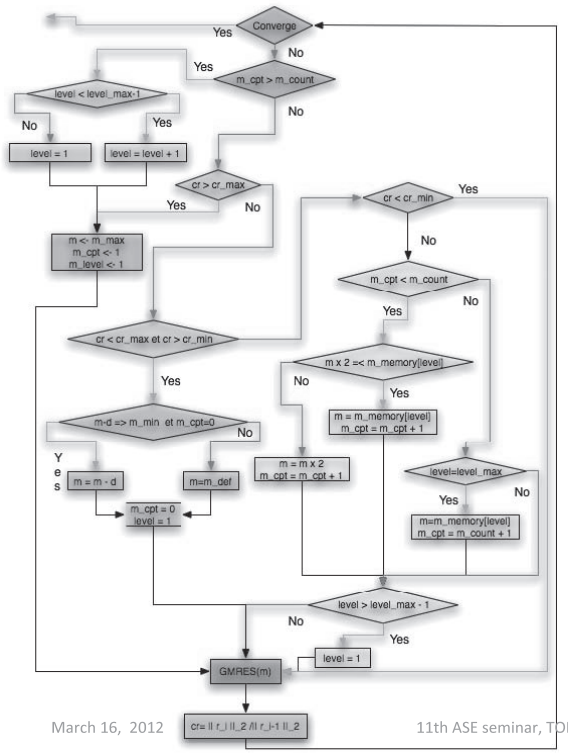
- Subspace size
  - Evaluate convergence progression over m iterations.
  - Decrease if convergence are monotonous or if they are smoothly slowing (approximately same convergence but minimize time and space)- **Cr medium (what value?)**
  - Increase if convergence stall (problem if we increase too much the memory space), Track memory levels : Cache, RAM, Nodes. **Cr low (what value?)**
  - Do nothing if **Cr high**(what value?)



Cr = norm2 (r_i) / norm2(r_{i_1}), Is it really a good criteria????

Do we have to change at each restart?

Easy to implement using libraries



Parameters

d : number of steps between successive decreases,

m_min: minimum subspace size value,

m_max : maximum subspace size value,

m_counts : number of successive classical increase before intending a more important one

m_memory[ ] : array containing subspace size values for important increase

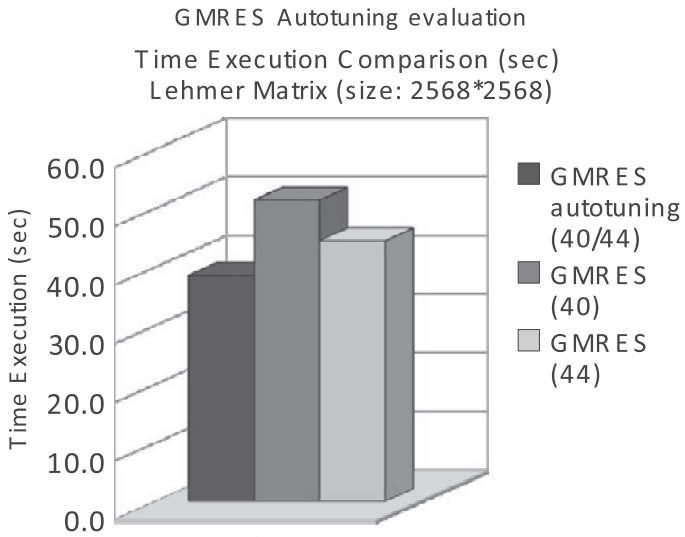
Cr-min : if <, then “Cr low”

Cr-max : if > then “Cr high”

GMRES Autotuning

Lehmer Matrix, size 2568*2568

GMRES(subspace size)



Outline

- Introduction
- Exascale challenges
- Krylov methods and auto-tuning algorithms
- **Incomplete orthogonalization auto-tuning algorithms**
- Experimental results
- Hybrid asynchronous krylov methods
- Towards smart-tuning and future intelligent numerical methods
- Conclusion

Incomplete orthogonalization Auto-Tuning

Complete orthogonalisation : we orthogonalise with all the previous computed vectors of the basis, i.e. at step j, we ortogonalise with j vectors, which generates j scalar product at step j. (j=1,m)

Incomplete orthogonalisation : we orthogonalize with only min(j,q) previous computed vectors of the basis, i.e. at step j, we ortogonalise only with min(j,q) vectors, q < m. DQGMRES : [Saad '94], DQGMRES : [Wu '97] IGMRES : [Brown '86][Jia '07]

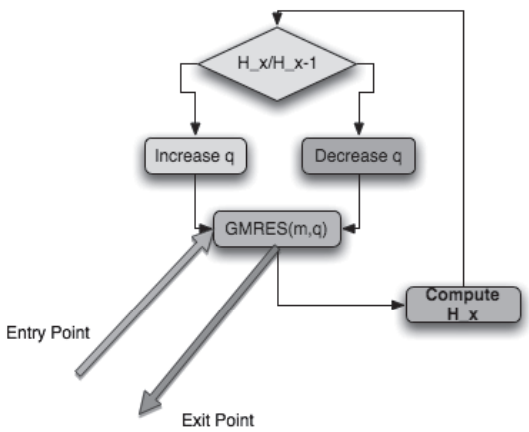
Then, we have only q scalar product at step j (for j > or equal to q).

Complete ortogonalization : j scalar product for j fixed ; 1,m
Incomplete orthogonalization : q scalar product for j fixed, q < j

We may then save j-q scalar products, for q < j, and, then, several synchronized communications .

Even, if the number of iterations may be a little larger, we minimize a lot of long blocking gobal communications generated by scalar products.

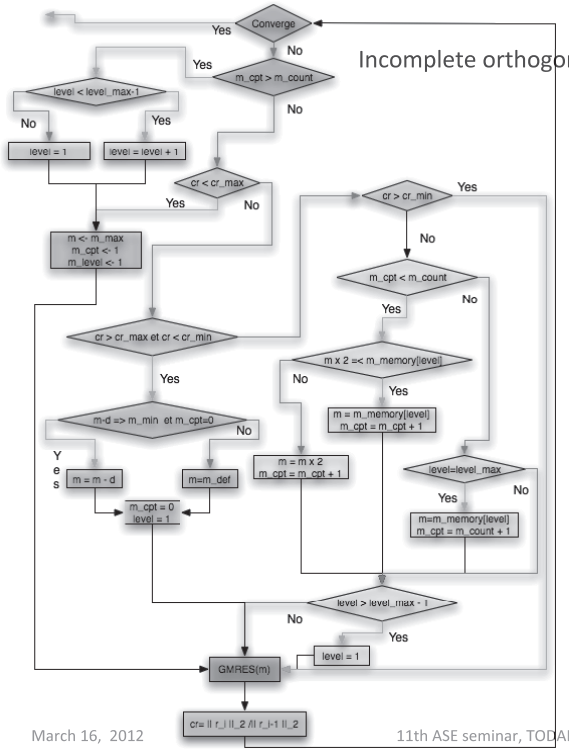
Incomplete orthogonalization algorithm at runtime



- Evaluate iteration costs in time vs. Convergence
- Decrease number of orthogonalized vectors q if ratio convergence/(time iteration) decrease

A complex heuristic-based algorithm : With respect to the variation of the residual between restarts, we change the number q of vectors concerned by the orthogonalization

Still, a lot of researches to achieve to optimize this algorithm.



Incomplete orthogonalization Auto-tuning Algorithm

q_{min}= minimum number of vector to orthogonalize

q_{max}= maximum number of vector to orthogonalize, typically = m the gmres subspace size

T_x = time of the x^{th} restart

N_x = norm of the residual variation, equal to the norm of the duration of the x^{th} restart minus the duration of the $x\text{-}1^{th}$ restart

H_x = relative variation
$= N_x / T_x$

Heuristic = ratio of the relative variation between restart x and x-1, equal H_x / H_{x-1}

GS, MGS,or GS with reorthogonalisation

- MGS is more stable but GS is more parallel,
- MGS have several scalar product in sequence,
- GS have a large granularity; matrice multiplication : n x k matrices, k = 1,m
- GS with systematic re-orthogonalization is a good compromise

- The sequential computation on the subspace is faster with incomplete orthogonalization as the projected Hessenberg matrix has just q+1 bands

- The incomplete orthogonalization is possible for all the orthogonalization processes,
- Nevertheless, MSG + incomplete auto-tuning has to be compared with GS with systematic orthogonalization

We already auto-tuned GMRES with respect to these different processes with complete orthogonaliszation.

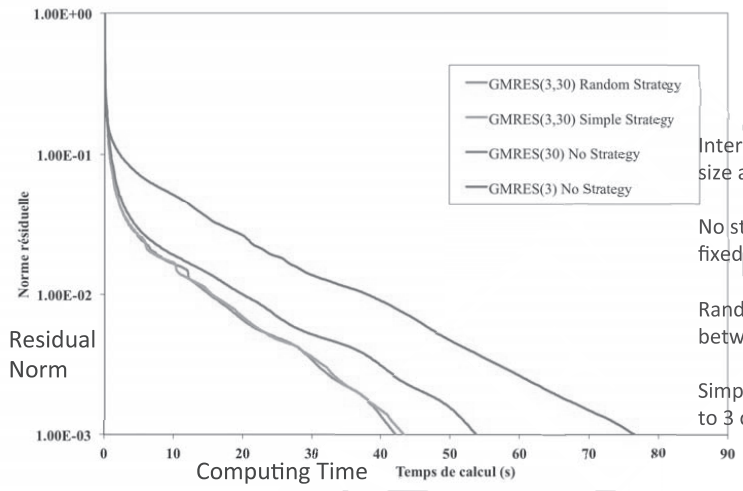
We'll evaluate these processes with the auto-tuning strategies

Outline

- Introduction
- Exascale challenges
- Krylov methods and auto-tuning algorithms
- Incomplete orthogonalization auto-tuning algorithms
- **Some experimental results**
- Hybrid asynchronous krylov methods
- Towards smart-tuning and future intelligent numerical methods
- Conclusion

Results : subspace size

- **Serial** : Auto-Tuning VS. No Auto-Tuning



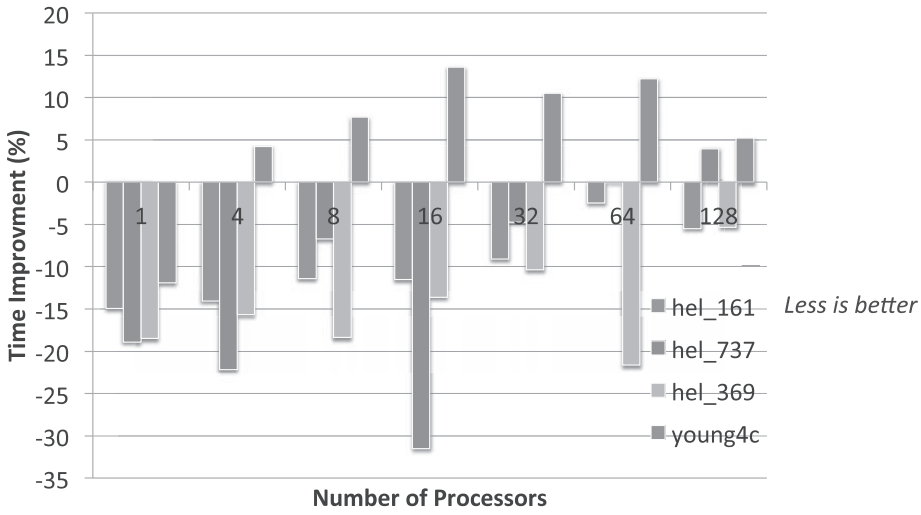
Interest of varying the subspace size at runtime

No strategy: m remain fixed for the full resolution

Random : m is a random value between 3 and 30

Simple : Decrease m from 30 to 3 cyclically

- Our algorithm compared to no auto-tuning



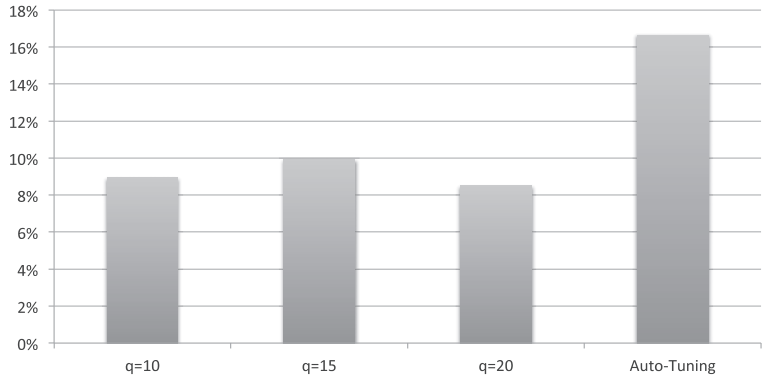
RESULTS Incomplete orthogonalization auto-tuning

Hardware : 2.26Ghz, Core2Duo, 4GB ram, PETSc 3.0

Matrix young4c from matrix market
Solution at 10^{e}-6
GMRES subspace size m=30
Truncation at 10, 15, 20
Serial processing

6 experiments for each case,
took the best time,

Percentage of improvement over full orthogonalisation
in term of computation time. Iteration number does not vary much



Higher is better

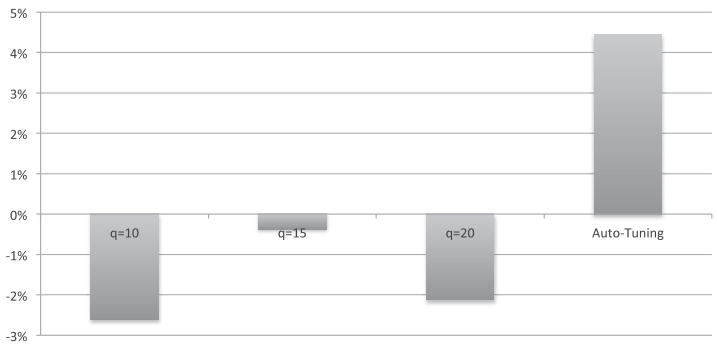
RESULTS Incomplete orthogonalization auto-tuning

Hardware : 2.26Ghz, Core2Duo, 4GB ram, PETSc 3.0

Matrix hel369 from matrix market
Solution at 10^{e}-3
GMRES subspace size m=30
Truncation at 10, 15, 20
Serial processing

6 experiments for each case,
took the best time

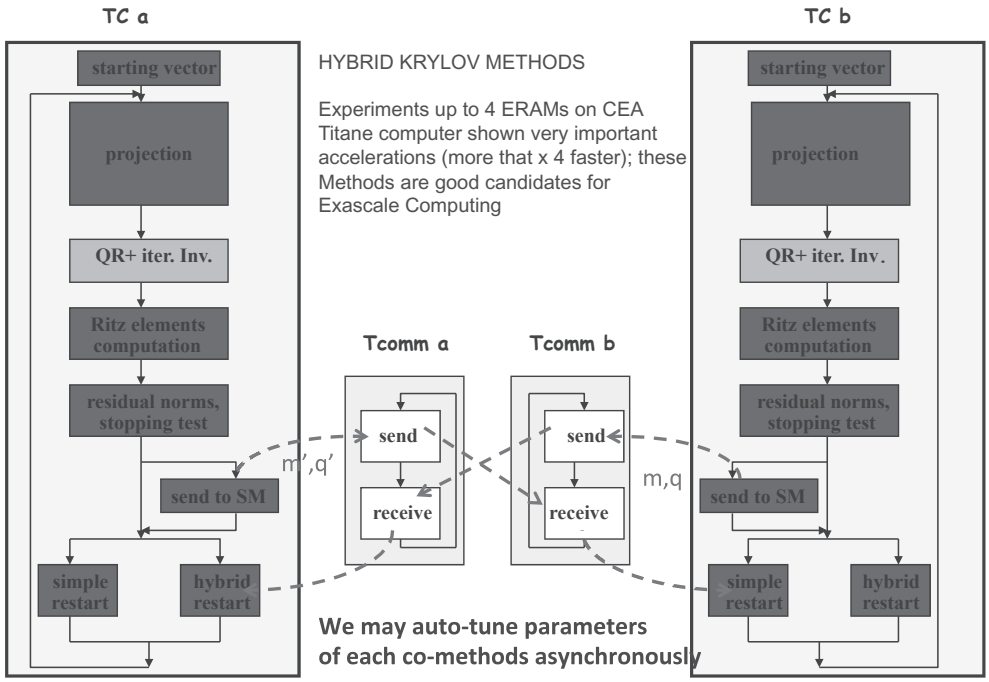
Percentage of improvement over full orthogonalisation
in term of computation time.



Higher is better

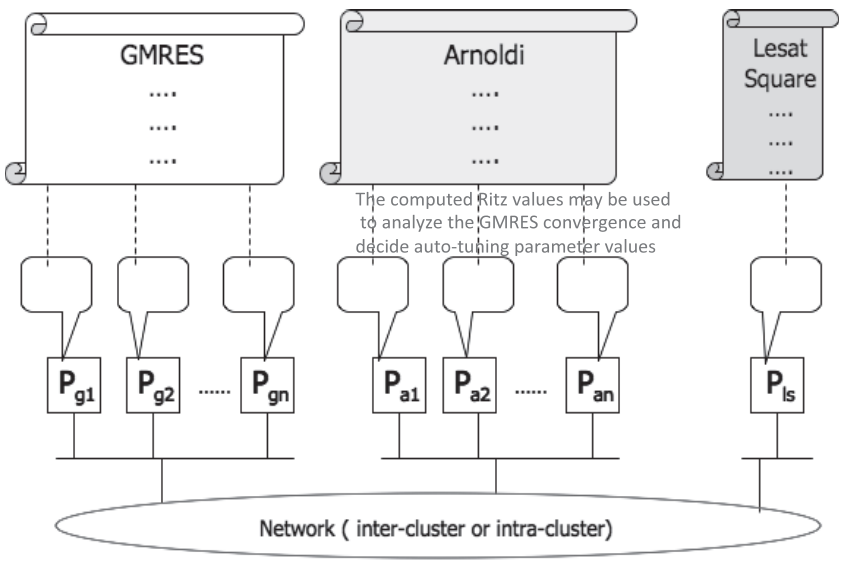
Outline

- Introduction
- Exascale challenges
- Krylov methods and auto-tuning algorithms
- Incomplete orthogonalization auto-tuning algorithms
- Experimental results
- **Hybrid asynchronous krylov methods**
- Towards smart-tuning and future intelligent numerical methods
- Conclusion



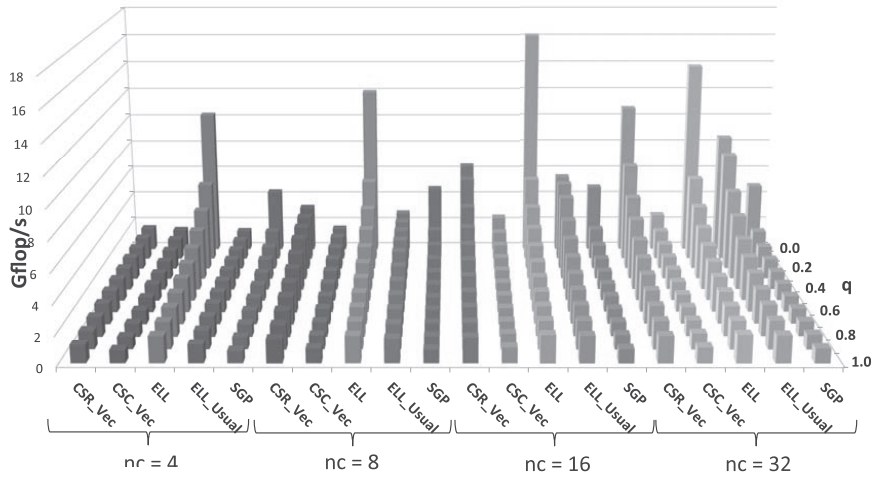
Asynchronous Iterative Restarted Methods

Collaboration with Guy Bergère and Ye Zhang



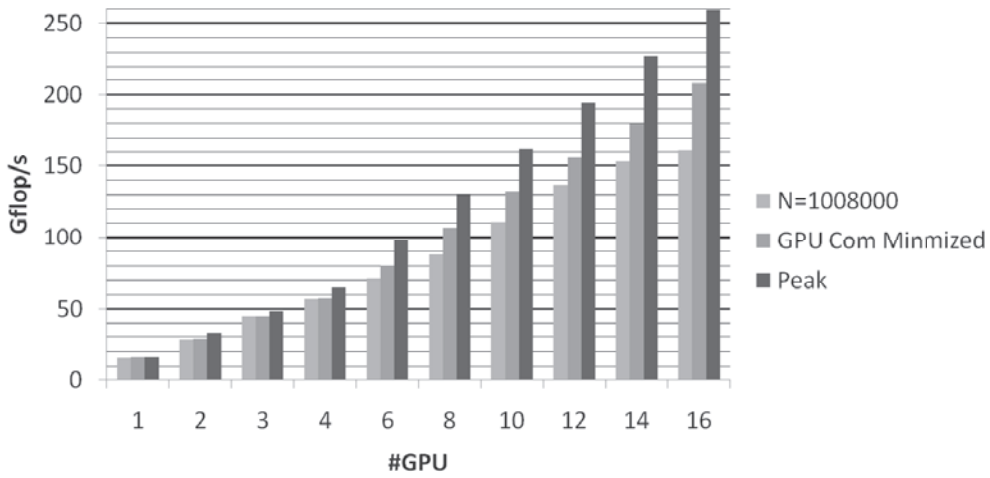
Performance of SpMV with C-Diagonal in Double Precision on a GPU

Important to auto-tune the compression format first



March 16, 2012 11th ASE seminar, TODAI

A(Ax) on Cluster of GPU with a larger matrix



Experiments with C-Diagonal
N=8064000 Q=0.0

March 16, 2012 11th ASE seminar, TODAI

Outline

- Introduction
- Exascale challenges
- Krylov methods and auto-tuning algorithms
- Incomplete orthogonalization auto-tuning algorithms
- Experimental results
- Hybrid asynchronous krylov methods
- **Towards smart-tuning and future intelligent numerical methods**
- Conclusion

- Each auto-tuning require to manage **several parameters**
- It is really very **difficult to analyze results on cluster of GPU** and or a large clusters of multi-cores, to many parameters are concerned, from architecture, software, data structures, auto-tuning algorithm, latencies,.....
- The different proposed auto-tuning techniques have interesting behaviours with respect of the matrices and the chose parameters ; but **it is impossible in the today state-of-the-art to really conclude that one is always the best**.
- We have to **include more numerical parameter**, such as Ritz elements for example, to be able to analyse the convergence at runtime and take decision about changing parameters,
- **Hybrid restarted methods** will have to asynchronously exchange auto-tuning parameter values to optimize local auto-tuning (ex MERAM(m1,m2,m3,…), GMRES/MERAM-LS)
- **Expertise from end-users** would be exploited through new high level language and/or framework (yml.prism.uvsq.fr); ex : cluster of eigenvalues,
- We have to analyse auto-tuned numerical methods to find **new criteria** to evaluate the quality of the converge and to decide actions

Outline

- Introduction
- Exascale challenges
- Krylov methods and auto-tuning algorithms
- Incomplete orthogonalization auto-tuning algorithms
- Experimental results
- Hybrid asynchronous krylov methods
- Towards smart-tuning and future intelligent numerical methods
- **Conclusion**

Collaborations are required

- The auto-tuning strategies have to become smarter and involve perhaps specific computations to allow smart-tuning
- On the road to Intelligente auto-tuned asynchronous hybrid restarted Krylov exascale methods!!!!!!