Balanced Truncation Model Reduction of Large and Sparse Generalized Linear Systems

Jos M. Badía[1], Peter Benner[2], Rafael Mayo[1], Enrique S. Quintana-Ortí[1], Gregorio Quintana-Ortí[1], A. Remón[1]

[1]Depto. de Ingeniería y Ciencia de Computadores
Universidad Jaume I de Castellón (Spain)
{badia,mayo,quintana,gquintan,remon}@icc.uji.es

[2]Fakultät für Mathematik
Technische Universität Chemnitz, Chemnitz (Germany)
benner@mathematik.tu-chemnitz.de

PMAA'06 - September 2006

Linear Systems

Generalized linear time-invariant systems:

$$\begin{aligned} E\dot{x}(t) &= Ax(t) + Bu(t), \quad t > 0, \quad x(0) = x^0, \\ y(t) &= Cx(t) + Du(t), \quad t \geq 0, \end{aligned}$$

- n state-space variables, i.e., n is the order of the system;
- m inputs,
- p outputs,
- $E^{-1}A$ is c-stable.

Corresponding TFM:

$$G(s) = C(sE - A)^{-1}B + D.$$

Model Reduction: Purpose

Given

$$\begin{aligned} E\dot{x}(t) &= Ax(t) + Bu(t), \quad t > 0, \quad x(0) = x^0, \\ y(t) &= Cx(t) + Du(t), \quad t \geq 0, \end{aligned}$$

find a reduced model

$$\begin{aligned} E_r\dot{x}_r(t) &= A_r x_r(t) + B_r u(t), \quad t > 0, \quad x_r(0) = x_r^0, \\ y_r(t) &= C_r x_r(t) + D_r u(t), \quad t \geq 0, \end{aligned}$$

of order $r \ll n$ and output error

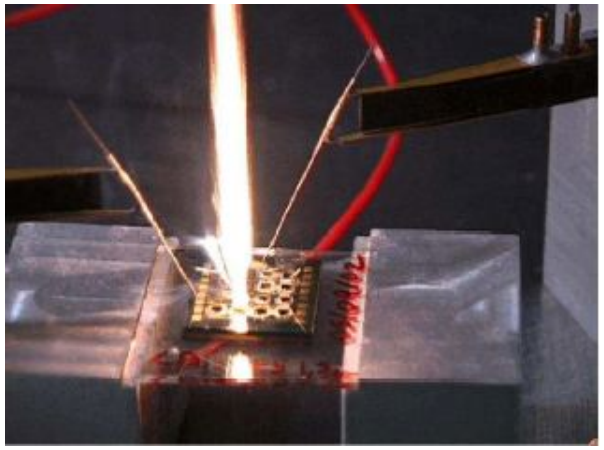
$$y - y_r = Gu - G_r u = (G - G_r)u$$

such that

$\|y - y_r\|$ and $\|G - G_r\|$ are "small"!

Model Reduction: MEMS Example

- Co-integration of solid fuel with silicon μ-machined system.
- Used for "nano-satellites" and gas generation.
- Design problem: reach the ignition temperature within the fuel without reaching the critical temperature at the neighbour μ-thrusters (boundary).



$n = 79{,}171$ states, $m = 1$ input, $p = 7$ outputs.

Outline

- Methods for model reduction: SRBT.
- Parallel solution of large-scale Lyapunov equations and other kernels.
- Analysis.
- Experimental results.
- Conclusions and future work.

Methods for Model Reduction

(Antoulas'02):

- Krylov-based approximation methods.
  - Numerically efficient and applicable to large-scale (sparse) systems.
- SVD-based approximation methods.
  - Preserve of stability.
  - Provide a global error bound on $\|G - G_r\|$.
  - Numerically efficient but applicable to large-scale (sparse) systems?

Balanced Truncation (Moore, 81)

Procedure composed of three steps:

1. Solve the "coupled" generalized Lyapunov matrix equations

$$\begin{aligned} AW_cE^T + EW_cA^T + BB^T &= 0, \\ A^T\hat{W}_oE + E^T\hat{W}_oA + C^TC &= 0, \end{aligned}$$

   for S, R such that $W_c = S^TS$, $W_o = E^T\hat{W}_oE = R^TR$.

2. Compute

$$SR^T = U\Sigma V^T = [\,U_1\ U_2\,] \begin{bmatrix} \Sigma_1 & \\ & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix},$$

   with $\Sigma_1 \in \mathbb{R}^{r\times r}$, $\Sigma_2 \in \mathbb{R}^{(n-r)\times(n-r)}$.

Balanced Truncation (Cont. I)

3. In the square-root method (Heath et al, 87; Tombs, Postlethwaite'87):

$$T_l = \Sigma_1^{-1/2} V_1^T R \quad \text{and} \quad T_r = S^T U_1 \Sigma_1^{-1/2},$$

and $(E_r, A_r, B_r, C_r, D_r) = (T_l E T_r, T_l A T_r, T_l B, C T_r, D)$.

- The Hankel singular values, $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n)$, provide a computable error bound:

$$\|G - G_r\|_\infty \leq 2 \sum_{k=r+1}^{n} \sigma_k.$$

This allows and adaptive choice of r.

Balanced Truncation (Cont. II)

Given (E, A, B, C, D) with E and A (sparse and) large, and $m, p \ll n$...

How do we solve/parallelize the previous numerical problems?

1. Coupled Lyapunov equations.
2. SVD of matrix product.
3. Application of the SR formulae to obtain the reduced-order model.

1. Solution of Coupled Lyapunov Equations

Generalization of the LR-ADI iteration in (Penzl, 98; Li, White, 99-02):

$$
\begin{aligned}
U_1 &= \gamma_1 \, (A + \tau_1 E)^{-1} B, \\
Y_1 &= U_1, \\
&\texttt{repeat } j = 2, 3, \ldots \\
&\qquad U_j = \gamma_j \left(U_{j-1} - (\tau_j + \overline{\tau_{j-1}})(A + \tau_j E)^{-1} E U_{j-1} \right), \\
&\qquad Y_j = [Y_{j-1}, U_j],
\end{aligned}
$$

where $\gamma_j = \sqrt{\mathrm{Re}(\tau_{j+1})/\mathrm{Re}(\tau_j)}$.

- $\tau = \{\tau_1, \tau_2, \ldots, \tau_s\}$ are the "shifts".
- Stop when contribution of U_j to Y_j is "small".
- Super-linear convergence.
- After l iterations, $W_c = S^T S \approx Y_l Y_l^T$, with $Y_l \in \mathbb{R}^{n \times (l \cdot m)}$.

1. Solution of Coupled Lyapunov equations (Cont. I)

Implementation:

- Set τ must be closed under complex conjugate computed by means of Arnoldi/Lanczos iterations.
- The LR-ADI iteration requires the solution of (sparse) linear systems with coefficient matrix E/A and matrix-vector product involving E/A.
- The iteration is applied cyclically: $\tau_{j+s} = \tau_l$
  $\rightarrow$ reuse factorization $(A + \tau_j E) = L_j U_j$ if available.
- Simplified formulation for symmetric definite pencils (E, A).
- Convergence criterion can be efficiently computed.
- Compression via RRQR possible.

1. Solution of Coupled Lyapunov equations (Cont. II)

Parallelization:

- Perform l iterations

$$U_j = \gamma_j \left(U_{j-1} - (\tau_j + \overline{\tau_{j-1}})(A + \tau_j E)^{-1} E U_{j-1} \right),$$

  with s different shifts.

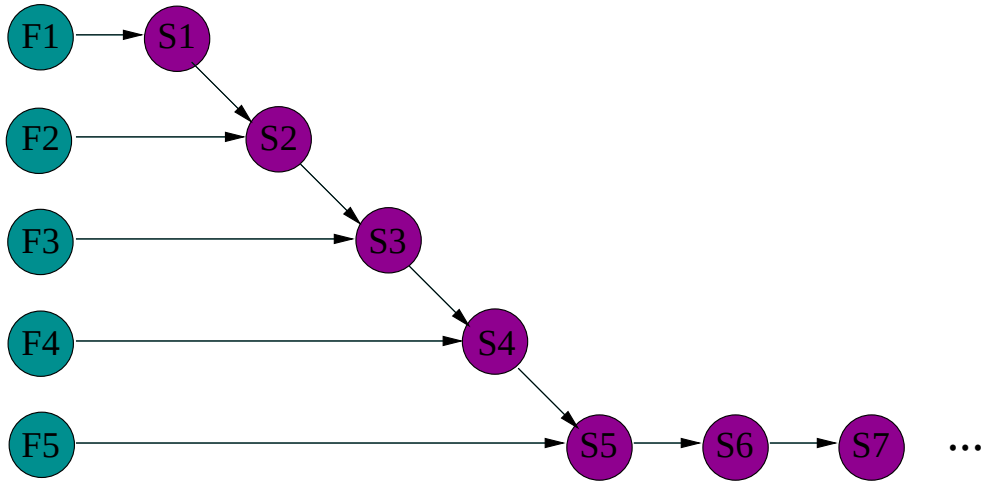
- There is little parallelism in solving each one of the systems in parallel.
- Given that l is usually of $\mathcal{O}(10)$ and $s \ll l$ a coarse-grain approach is feasible

1. Solution of Coupled Lyapunov equations (Cont. III)

Consider two different classes of tasks:

- F_k: Factorize $(A + \tau_k E)$, $k = 1, 2, \ldots, s$.
- S_k: Solve $(A + \tau_k E)^{-1}(EU_{k-1})$, $k = 1, 2, \ldots, l$.

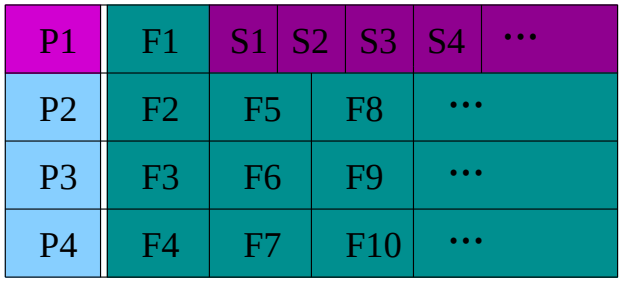
The data dependencies graph (for $s = 5$ factorizations) is:



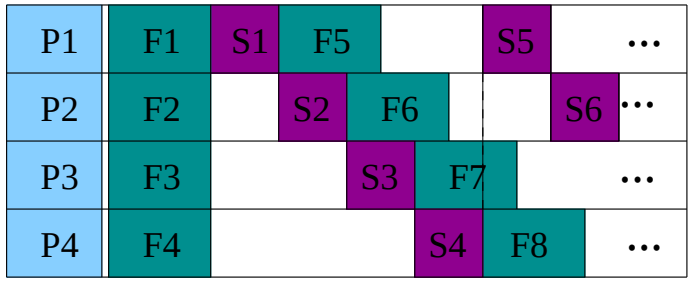
1. Solution of Coupled Lyapunov equations (Cont. IV)

Two different coarse-grain parallel algorithmic schemes:

Heterogeneous scheme



Homogeneous scheme



- Heterogeneous (HT) scheme: All factorization pass through P_1 allowing a producers/consumer (P2,P3,P4/P1) scheme.
- Homogeneous (HM) scheme: Solutions move among processors resulting in an homogeneous role for all computational resources.

1. Solution of Coupled Lyapunov equations (Cont. V)

Two different coarse-grain parallel implementations:

- Multithreaded (MT) implementation. One thread per processor.
  All data must be shared.
- Multiprocess (MP) implementation. One process per processor: message-passing.
  Data is communicated via MPI or shared-memory zones.

Efficiency of solutions HT-MT, HT-MP, HM-MT, HM-MP depends on:

- Values of s (selected by user) and l (not known *a priori*).
- Ratio of factorization vs. solution execution times.
- Number of resources.
- Performance of communication mechanism.

2. SVD of Matrix Product

Replace the Cholesky factors by their low-rank approximations in

$$SR^T \approx \hat{S}_l^T \hat{R}_l = U\Sigma V^T.$$

Implementation:

- The product $\hat{S}_l^T \hat{R}_l$ is of order $(l \cdot m) \times (l \cdot p)$, m, $p \ll n$,
  $\longrightarrow$ use dense LA methods.
- Accuracy can be enhanced by computing the SVD without computing explicitly the product, but it is difficult to parallelize.

Parallelization:

- ScaLAPACK.

3. Application of SRBT Formulae

Implementation:

- Computation of the projection matrices

$$\begin{aligned} T_l &= \Sigma_1^{-1/2} V_1^T \hat{R}_k^T, \\ T_r &= \hat{S}_k U_1 \Sigma_1^{-1/2}, \end{aligned}$$

  only requires dense LA methods.

- Computation of the reduced-order matrices

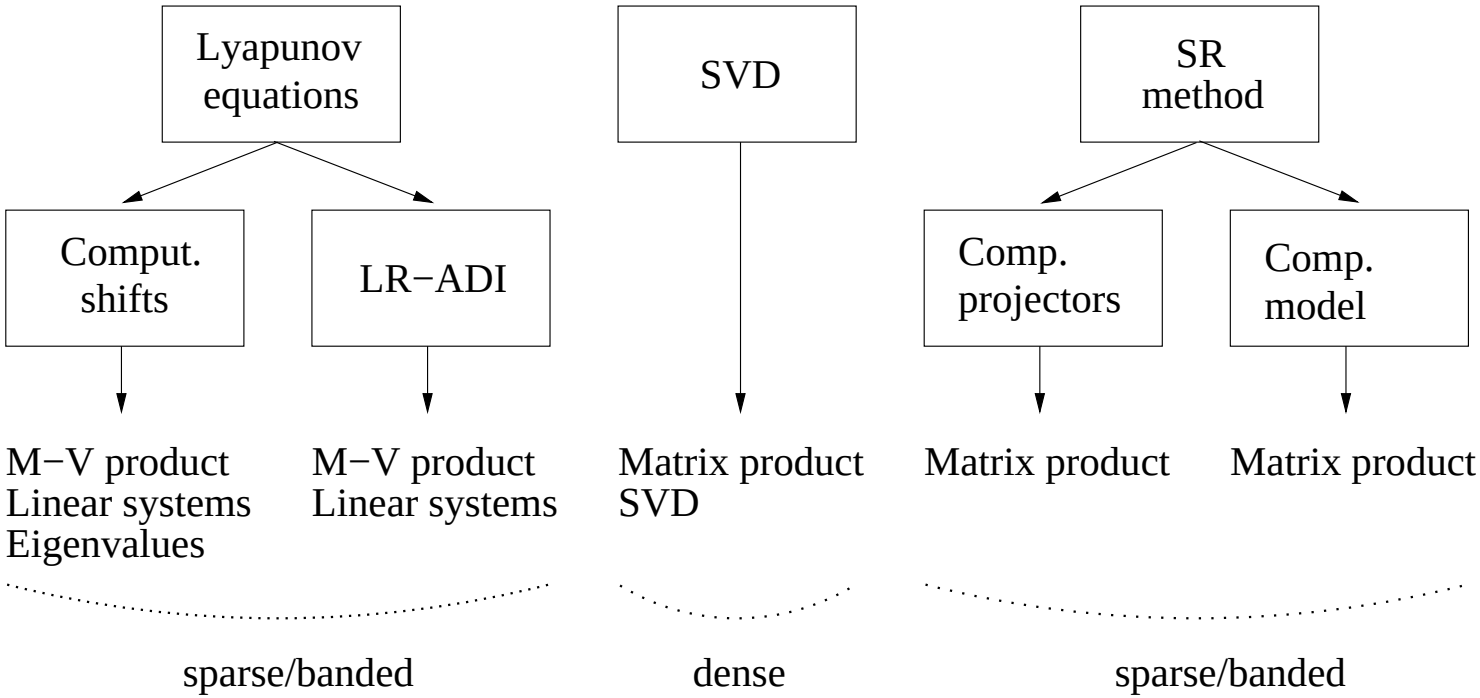
$$\begin{aligned} E_r &= T_l E T_r = I_r, & A_r &= T_l A T_r, \\ B_r &= T_l B, & C_r &= C T_r, \end{aligned}$$

  can be easily done exploiting any sparsity/pattern in A, B, or C.

Parallelization:

- Parallel implementation of sparse matrix product.

Summary of model reduction procedure



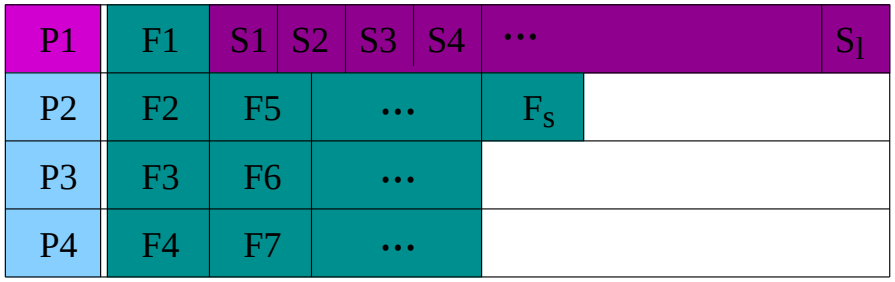
The main computational cost is in the LR-ADI iteration!

Analysis

- Given factorization and solution times T_F and T_S, resp., the sequential time is

$$T_{seq} = T_F \cdot s + T_S \cdot l$$

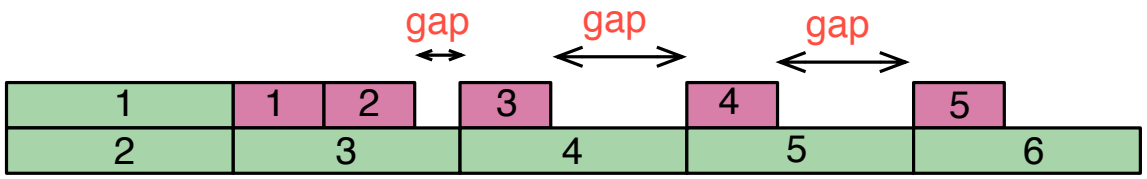
- At best, all factorizations but the first can be overlapped with the (sequential) solutions:



$$T_{par} = T_F + T_S \cdot l$$

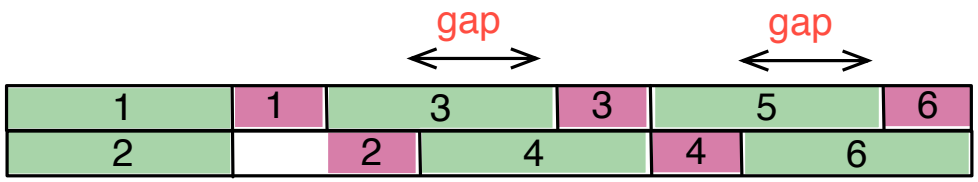
Analysis (Cont. I)

- Heterogeneous scheme:



$$T_{par}^{HT} = T_{par} + \max(0, T_F - T_S(n_p - 1)) \left(\lceil s/n_p \rceil - 1\right)$$

- Homogeneous scheme:



$$T_{par}^{HM} = T_{par} + \max(0, \lceil (s - n_p)/(n_p - 1) \rceil \cdot \max(0, T_F - T_S(n_p - 1)) - T_S)$$

with the minimum attained for $n_p \geq \lceil \frac{T_F}{T_S} \rceil + 1$ in both schemes.

Experimental Results

Computing facility:

- SGI Altix 350: 7 nodes $\times$ 2 Intel Itanium2 processors@1.5GHz, with 30 GB of shared RAM connected via a SGI NUMAlink
- IEEE double precision arithmetic.
- LAPACK and BLAS in MKL 8.1
- Sparse solver in MUMPS 4.6.2

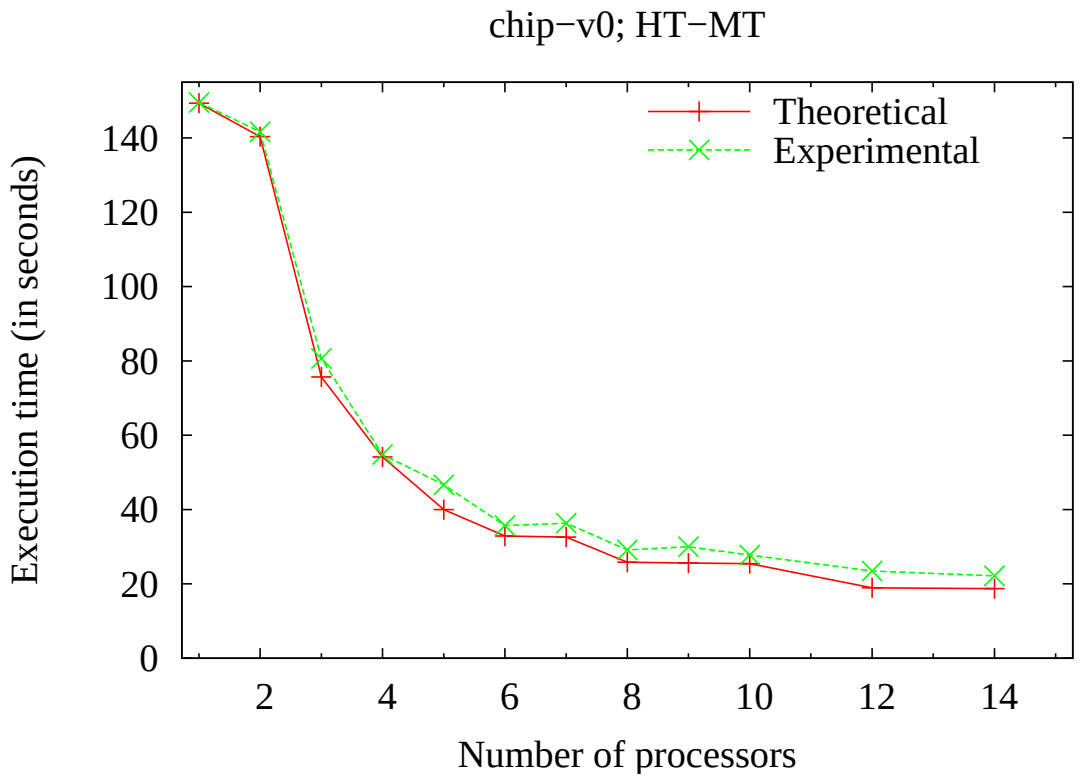
Experimental Results (Cont. I)

Testbed: Oberwolfach model reduction benchmark collection (`http://www.imtek.uni-freiburg.de/simulation/benchmark/`)

- Convective thermal flow problems: `chip_v0`.
- Micropyros thruster: `t3dl`.
- Semi-discretized heat transfer problem for optimal cooling of steel profiles: `rail_20209`.

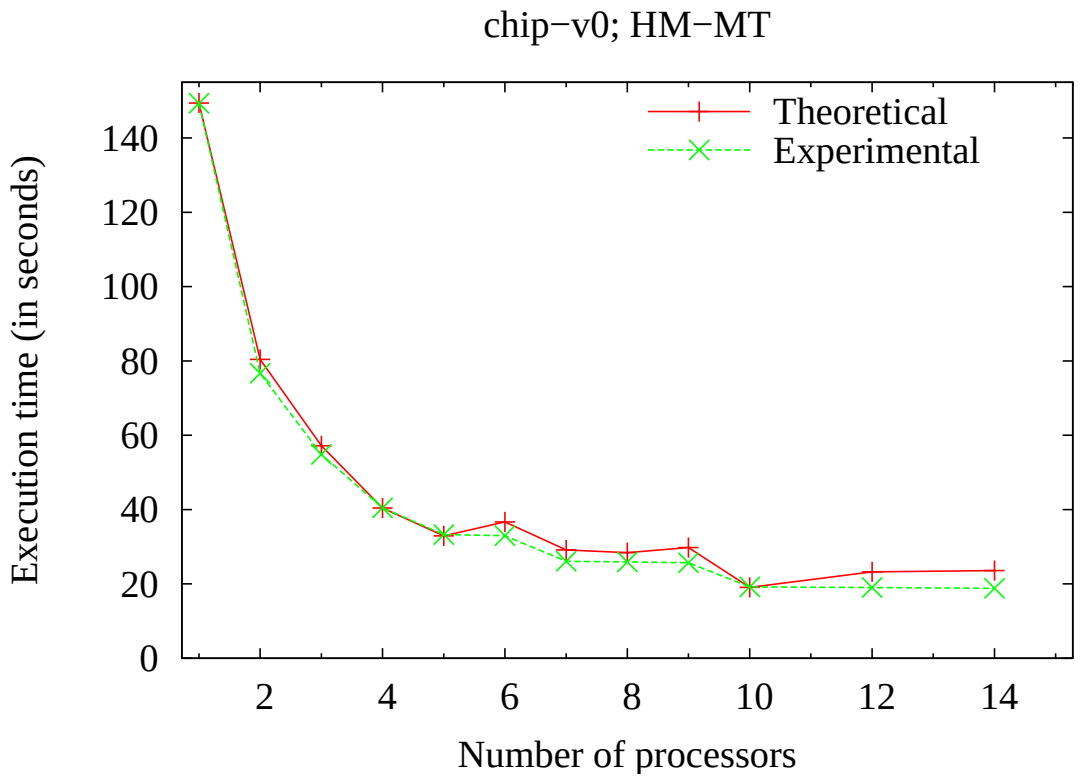
Experimental Results (Cont. II)

Comparison of theoretical versus experimental data



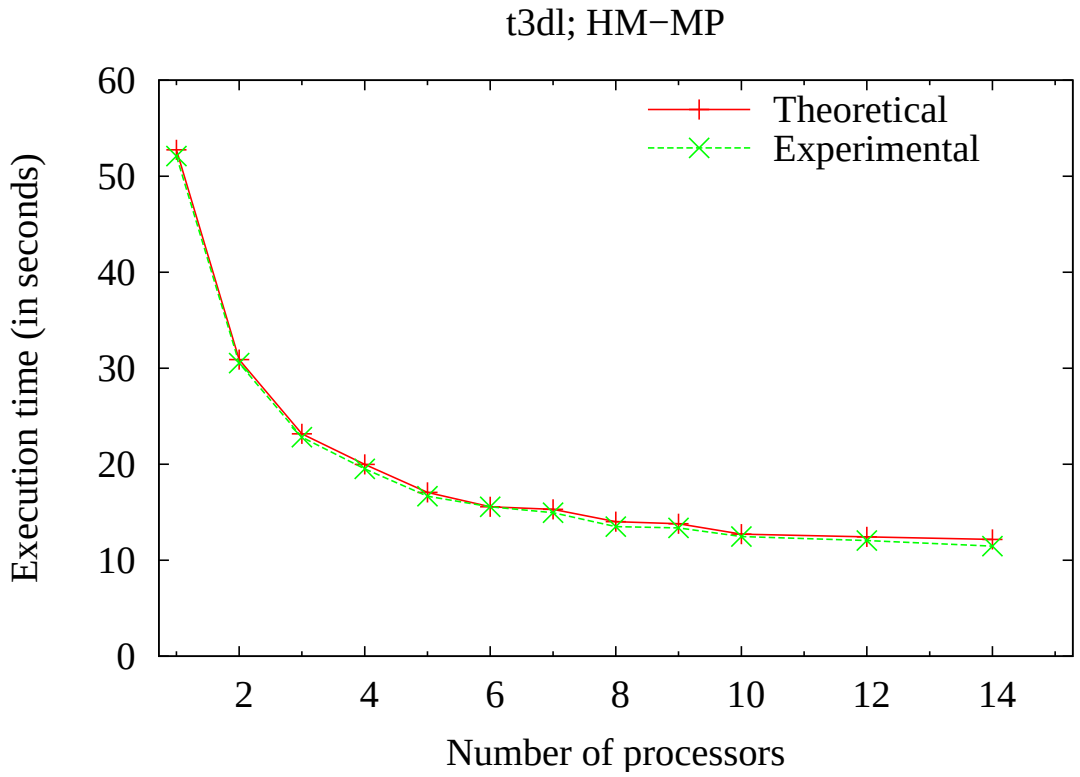
Experimental Results (Cont. III)

Comparison of theoretical versus experimental data



Experimental Results (Cont. IV)

Comparison of theoretical versus experimental data



Experimental Results (Cont. V)

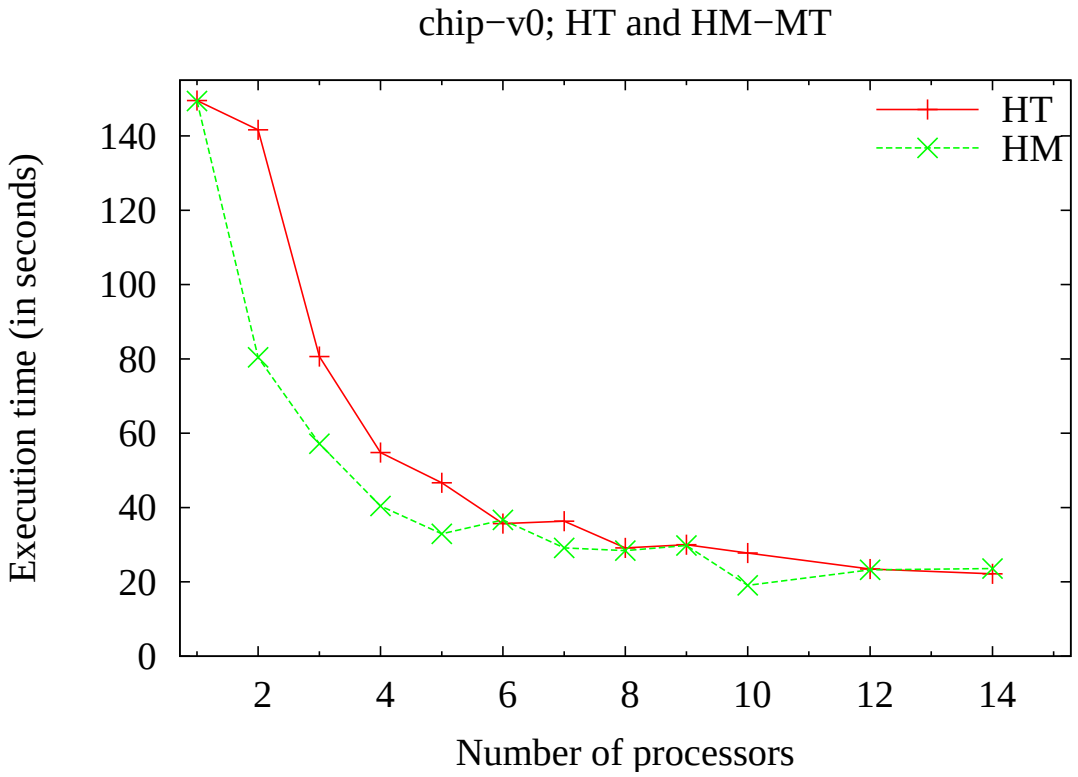
Comparison of theoretical versus experimental data

Heterogeneous scheme and multiprocess implementation?

Requires passing of data structures corresponding to (sparse) factorizations between processes $\longrightarrow$ hidden under sparse solver

Experimental Results (Cont. V)

Comparison of HM versus HT schemes



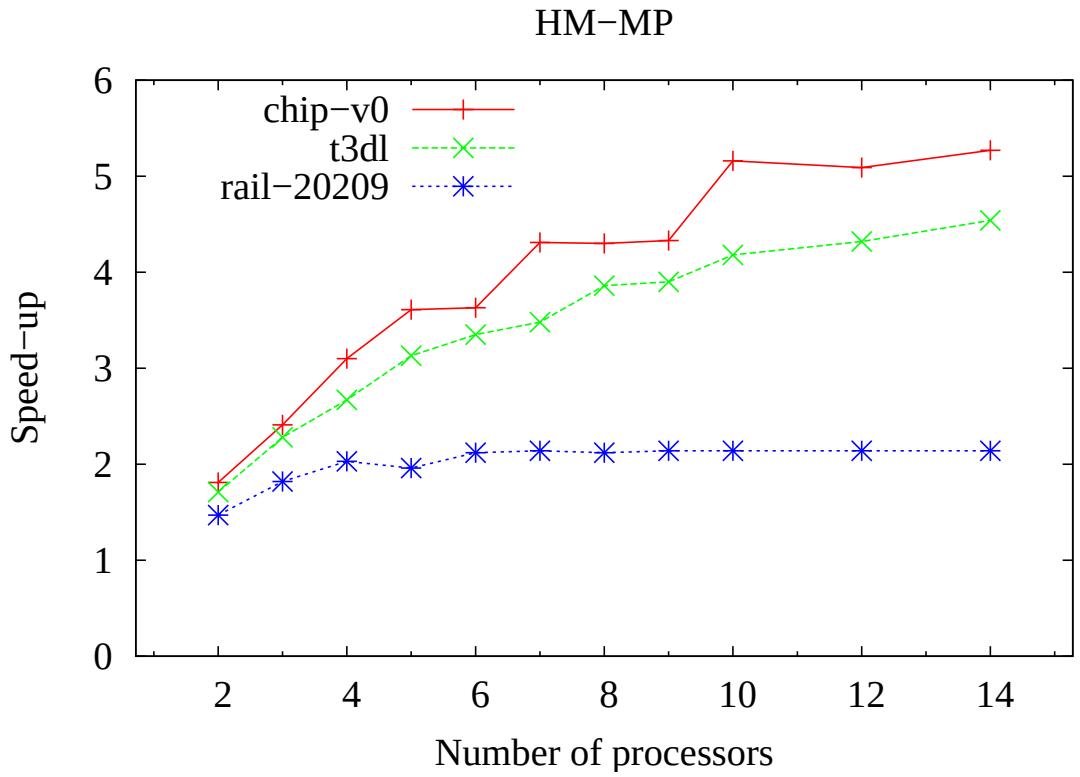
Experimental Results (Cont. VI)

What can be expected for the speed-up of HT?

Example	s	T_F	l	T_S	T_{seq}	$\min(T_{par})$	$\max(S_p)$
`chip_v0`	20	1.10	59	0.06	25.74	4.85	5.29
`t3dl`	29	1.49	140	0.07	52.75	10.94	4.82
`rail_20209`	25	0.35	78	0.09	5.88	7.60	2.09

Experimental Results (Cont. VII)

Speed-up of HM-MP



Concluding Remarks

- Parallel SRBT algorithms in SpaRed allow reduction of sparse systems with $\mathcal{O}(10^5)$ states.
- Coarse-grain parallelization is possible, in general with better performances than using parallel implementations of sparse linear system solvers (MUMPS, SuperLU, etc.)
- Two schemes and two implementations $\longrightarrow$ four variants.
- Theoretical models match experimental results with high accuracy.
- Parallel performance strongly depends on the problem. Different cases present extreme dissimilar values for T_F, T_S, l, s,...
- HM is better than HT when n_p is small. As n_p increases, the efficacy of both models tends to be equal.

Future Work

- Improve serial codes for factorization and solution: band codes in LAPACK. Sparse matrix-vector product.
- Combine MP and MT in a hybrid algorithm: Several threads to solve concurrently the triangular linear systems (the bottleneck of the parallel algorithm).
- Experiment with other sparse solvers: WSMP, UMFPACK, etc.

Thank you for your attention

Questions?