Algorithm-based checkpoint-recovery for the conjugate gradient method

Carlos Pachajoa, Christina Pacher, Markus Levonyak, Wilfried N. Gansterer.

49th International Conference on Parallel Processing

universität wien

Acknowledgements

This work has been funded by the Vienna Science and Technology Fund through project ICT15-113.

Experiments are run on the VSC3 machine of the Vienna Scientific Cluster.

Motivation

1. Unreliability at larger scales
   - Reliability of larger-scale computer systems is predicted to decline.
   - Computers can no longer be thought as reliable machines. Resilience is an active research field.
   - We focus on node failures: Events in which a node stops working and the data contained in it is lost. Several nodes can fail simultaneously if, for example, a switch stops working.
2. Resilience for the conjugate gradient method
   - Iterative solver for symmetric, positive definite (SPD) linear systems.
   - Significant in many physically-motivated problems.
   - Particularly suitable for work with sparse matrices. Usable in very large systems.

Problem statement

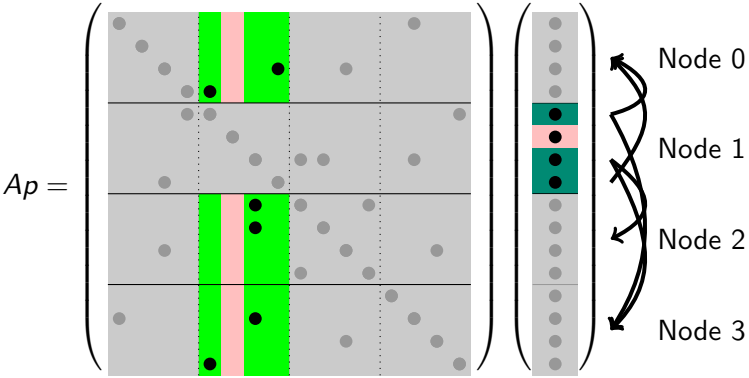
- Unreliable computer cluster. Possibility of node failures occuring.
- Find the solution of a linear system for an SPD matrix using the conjugate gradient method.
- Sparse matrices stored with a block-row distribution. Vector elements distributed in the same way as the rows.

Key idea 1: Matrix-vector product

- The matrix-vector product provides some redundancy for the input vector, and can be augmented to guarantee complete redundancy.

Storing redundantly for a node failure

The matrix-vector product provides some redundancy for the input vector (Chen 2011). In this example, we focus on the second rank: One of its entries is not necessary for the SpMV and must be sent additionally.

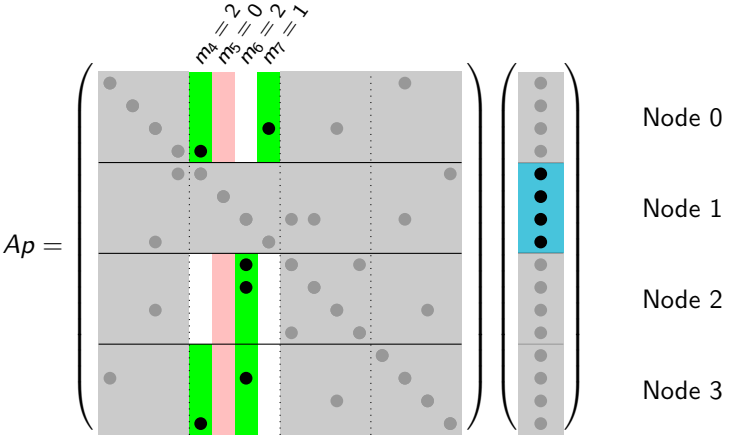


Multiple node failures

- These ideas can be generalized to deal with multiple, simultaneous node failures, for example, in the event of a switch failure.

Augmented SpMV product for multiple node failures

m_i: Multiplicity of entry i.



To guarantee we can recover from up to ϕ node failures, the SpMV must be augmented until the multiplicity for every entry of each node is $m_i \geq \phi$

Key idea 2: State reconstruction

The complete state can be recovered from the two last *search directions* (the $\boldsymbol{p}$ vector) (Pachajoa et al. 2019).

Preconditioned conjugate gradient method

$\boldsymbol{r}^{(0)} := \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(0)}, \boldsymbol{z}^{(0)} := \boldsymbol{P}\boldsymbol{r}^{(0)}, \boldsymbol{p}^{(0)} := \boldsymbol{z}^{(0)}$

repeat

  $\alpha^{(j)} := \boldsymbol{r}^{(j)\top}\boldsymbol{z}^{(j)} / \boldsymbol{p}^{(j)\top}\boldsymbol{A}\boldsymbol{p}^{(j)}$;

  $\boldsymbol{x}^{(j+1)} := \boldsymbol{x}^{(j)} + \alpha^{(j)}\boldsymbol{p}^{(j)}$;

  $\boldsymbol{r}^{(j+1)} := \boldsymbol{r}^{(j)} - \alpha^{(j)}\boldsymbol{A}\boldsymbol{p}^{(j)}$;

  $\boldsymbol{z}^{(j+1)} := \boldsymbol{P}\boldsymbol{r}^{(j+1)}$;

  $\beta^{(j)} := \boldsymbol{r}^{(j+1)\top}\boldsymbol{z}^{(j+1)} / \boldsymbol{r}^{(j)\top}\boldsymbol{z}^{(j)}$;

  $\boldsymbol{p}^{(j+1)} := \boldsymbol{z}^{(j+1)} + \beta^{(j)}\boldsymbol{p}^{(j)}$;

until $\|\boldsymbol{r}\|_2 / \|\boldsymbol{b}\|_2 < rtol$;

Exact state reconstruction method

Retrieve the static data $\boldsymbol{A}_{I_f,I}$, $\boldsymbol{P}_{I_f,I}$, and $\boldsymbol{b}_{I_f}$;

Gather $\boldsymbol{r}^{(j)}_{I\setminus I_f}$ and $\boldsymbol{x}^{(j)}_{I\setminus I_f}$;

Retrieve the redundant copies of $\beta^{(j-1)}$, $\boldsymbol{p}^{(j-1)}_{I_f}$, and $\boldsymbol{p}^{(j)}_{I_f}$;

Compute $\boldsymbol{z}^{(j)}_{I_f} := \boldsymbol{p}^{(j)}_{I_f} - \beta^{(j-1)}\boldsymbol{p}^{(j-1)}_{I_f}$;

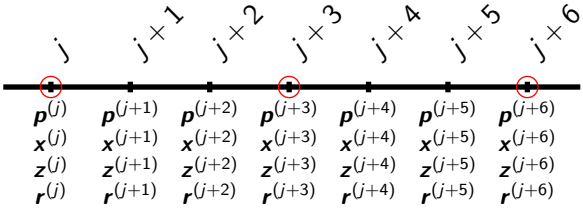
Compute $\boldsymbol{v} := \boldsymbol{z}^{(j)}_{I_f} - \boldsymbol{P}_{I_f,I\setminus I_f}\boldsymbol{r}^{(j)}_{I\setminus I_f}$;

Solve $\boldsymbol{P}_{I_f,I_f}\boldsymbol{r}^{(j)}_{I_f} = \boldsymbol{v}$ for $\boldsymbol{r}^{(j)}_{I_f}$;

Compute $\boldsymbol{w} := \boldsymbol{b}_{I_f} - \boldsymbol{r}^{(j)}_{I_f} - \boldsymbol{A}_{I_f,I\setminus I_f}\boldsymbol{x}^{(j)}_{I\setminus I_f}$;

Solve $\boldsymbol{A}_{I_f,I_f}\boldsymbol{x}^{(j)}_{I_f} = \boldsymbol{w}$ for $\boldsymbol{x}^{(j)}_{I_f}$;

Key idea 3: Reduce the overhead by storing every T iterations



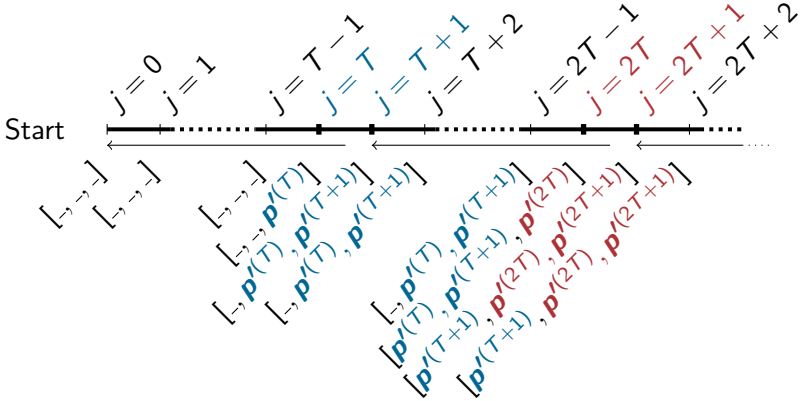
ESR algorithm

1. Retrieve the static data $\boldsymbol{A}_{I_f,I}$, $\boldsymbol{P}_{I_f,I}$, and $\boldsymbol{b}_{I_f}$;
2. Gather $\boldsymbol{r}^{(j)}_{I\setminus I_f}$ and $\boldsymbol{x}^{(j)}_{I\setminus I_f}$;
3. Retrieve the redundant copies of $\beta^{(j-1)}$, $\boldsymbol{p}^{(j-1)}_{I_f}$, and $\boldsymbol{p}^{(j)}_{I_f}$;
4. Compute $\boldsymbol{z}^{(j)}_{I_f} := \boldsymbol{p}^{(j)}_{I_f} - \beta^{(j-1)}\boldsymbol{p}^{(j-1)}_{I_f}$;
5. Compute $\boldsymbol{v} := \boldsymbol{z}^{(j)}_{I_f} - \boldsymbol{P}_{I_f,I\setminus I_f}\boldsymbol{r}^{(j)}_{I\setminus I_f}$;
6. Solve $\boldsymbol{P}_{I_f,I_f}\boldsymbol{r}^{(j)}_{I_f} = \boldsymbol{v}$ for $\boldsymbol{r}^{(j)}_{I_f}$;
7. Compute $\boldsymbol{w} := \boldsymbol{b}_{I_f} - \boldsymbol{r}^{(j)}_{I_f} - \boldsymbol{A}_{I_f,I\setminus I_f}\boldsymbol{x}^{(j)}_{I\setminus I_f}$;
8. Solve $\boldsymbol{A}_{I_f,I_f}\boldsymbol{x}^{(j)}_{I_f} = \boldsymbol{w}$ for $\boldsymbol{x}^{(j)}_{I_f}$;

Two problems:

- To use the reconstructed parts, we also need the corresponding entries of the vector for that iteration.
  - Therefore, all nodes must store their local parts of the vector at the checkpoint.
- We need $\boldsymbol{p}$ for two consecutive iterations to be able to perform the reconstruction.
  - We need a queue of redundantly stored data.

Storing redundant data every few iterations



Definition of ASpMV

- The function `SpMV` takes a matrix and a vector as inputs, and outputs a vector.
- The function `ASpMV` additionally takes a target number of redundant copies (ϕ) and a queue to store them (Q).

$$\boldsymbol{\varrho} := \mathtt{SpMV}(\boldsymbol{A}, \boldsymbol{p})$$

$$\boldsymbol{\varrho} := \mathtt{ASpMV}(\boldsymbol{A}, \boldsymbol{p}, \phi, Q)$$

Reducing the frequency: CG-ESRP

Preconditioned conjugate gradient method

$\boldsymbol{r}^{(0)} := \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(0)}, \boldsymbol{z}^{(0)} := \boldsymbol{P}\boldsymbol{r}^{(0)}, \boldsymbol{p}^{(0)} := \boldsymbol{z}^{(0)}$
repeat
  $\alpha^{(j)} := \boldsymbol{r}^{(j)\top}\boldsymbol{z}^{()}/\boldsymbol{p}^{(j)\top}\boldsymbol{A}\boldsymbol{p}^{(j)};$
  $\boldsymbol{x}^{(j+1)} := \boldsymbol{x}^{(j)} + \alpha^{(j)}\boldsymbol{p}^{(j)};$
  $\boldsymbol{r}^{(j+1)} := \boldsymbol{r}^{(j)} - \alpha^{(j)}\boldsymbol{A}\boldsymbol{p}^{(j)};$
  $\boldsymbol{z}^{(j+1)} := \boldsymbol{P}\boldsymbol{r}^{(j+1)};$
  $\beta^{(j)} := \boldsymbol{r}^{(j+1)\top}\boldsymbol{z}^{(j+1)}/\boldsymbol{r}^{(j)\top}\boldsymbol{z}^{(j)};$
  $\boldsymbol{p}^{(j+1)} := \boldsymbol{z}^{(j+1)} + \beta^{(j)}\boldsymbol{p}^{(j)};$
until $\|\boldsymbol{r}\|_2/\|\boldsymbol{b}\|_2 < rtol;$

Conjugate gradient method using exact state reconstruction with periodic storage (CG-ESRP)

$\boldsymbol{r}^{(0)} := \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(0)}, \boldsymbol{z}^{(0)} := \boldsymbol{P}\boldsymbol{r}^{(0)}, \boldsymbol{p}^{(0)} := \boldsymbol{z}^{(0)}, j := 0,$
$Q := [_, _, _];$
repeat
  if $j \ mod \ T = 0 \ \ and \ \ j > 2$ **then**
    $\boldsymbol{\varrho}^{(j)} := \mathtt{ASpMV}(\boldsymbol{A}, \boldsymbol{p}^{(j)}, \phi, Q);$
    $\beta ** = \beta^{(j)};$
  else if $(j-1) \ mod \ T = 0 \ \ and \ \ j > 2$ **then**
    $\boldsymbol{\varrho}^{(j)} := \mathtt{ASpMV}(\boldsymbol{A}, \boldsymbol{p}^{(j)}, \phi, Q);$
    $\boldsymbol{x}* = \boldsymbol{x}^{(j)}, \ \boldsymbol{r}* = \boldsymbol{r}^{(j)}, \ \boldsymbol{z}* = \boldsymbol{z}^{(j)}, \ \boldsymbol{p}* = \boldsymbol{p}^{(j)};$
    $\beta* = \beta **;$
  else
    $\boldsymbol{\varrho}^{(j)} := \mathtt{SpMV}(\boldsymbol{A}, \boldsymbol{p}^{(j)});$
  $\alpha^{(j)} := \boldsymbol{r}^{(j)\top}\boldsymbol{z}^{(j)}/\boldsymbol{p}^{(j)\top}\boldsymbol{\varrho}^{(j)};$
  $\boldsymbol{x}^{(j+1)} := \boldsymbol{x}^{(j)} + \alpha^{(j)}\boldsymbol{p}^{(j)};$
  $\boldsymbol{r}^{(j+1)} := \boldsymbol{r}^{(j)} - \alpha^{(j)}\boldsymbol{\varrho}^{(j)};$
  $\boldsymbol{z}^{(j+1)} := \boldsymbol{P}\boldsymbol{r}^{(j+1)};$
  $\beta^{(j)} := \boldsymbol{r}^{(j+1)\top}\boldsymbol{z}^{(j+1)}/\boldsymbol{r}^{(j)\top}\boldsymbol{z}^{(j)};$
  $\boldsymbol{p}^{(j+1)} := \boldsymbol{z}^{(j+1)} + \beta^{(j)}\boldsymbol{p}^{(j)};$
  $j := j + 1;$
until $\|\boldsymbol{r}\|_2/\|\boldsymbol{b}\|_2 < rtol;$

Experimental setup

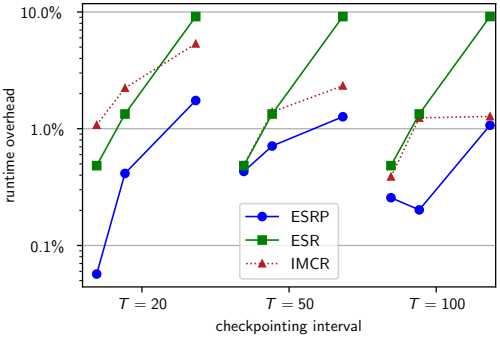
- 128 nodes of the VSC3.
- Two strategies to recover: ESRP and in-memory CR (IMCR).
- Simulated node failures.
- Checkpointing interval of 20, 50 and 100 iterations.
- Resilience with 1, 3 and 8 redundant copies.
- Runs without resilience, and with resilience with and without node failures.

Test matrices from the SuiteSparse collection (Davis and Hu 2011)

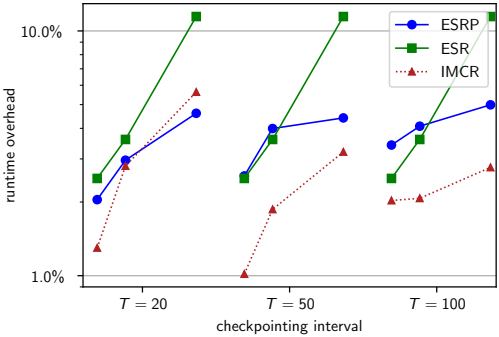
Matrix	**Problem type**	**Problem size**	**#NZ**
Emilia_923	Structural	923 136	40 373 538
audikw_1	Structural	943 695	77 651 847

Results for matrix Emilia_923

- Reference time $t_0 = 14.66s$
- σ_{t_0} is 0.93% of t_0.



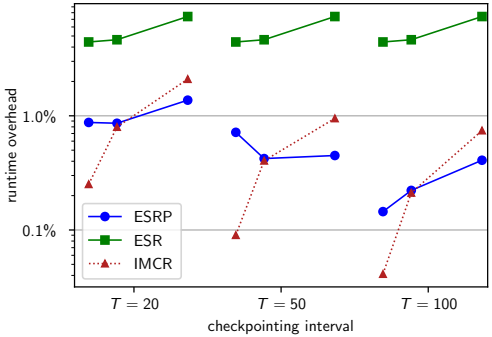
(a) Failure-free solver



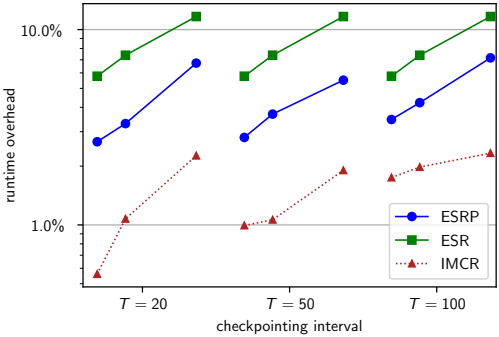
(b) Node failures introduced

Results for matrix audikw_1

- Reference time $t_0 = 23.22s$
- σ_{t_0} is 0.14% of t_0.



(a) Failure-free solver



(b) Node failures introduced

Conclusions and perspectives

Conclusions

- In our first experiments, ESRP drastically reduces the overhead of ESR.
- In failure-free cases, also faster than in-memory CR.
- In our experiments, the cost of communication seems to be too low. We cannot conclude that IMCR is faster than ESRP in this case.
- Recovery time for ESRP is dominated by the solution of the local linear system during reconstruction.

Perspectives

- Experiments with larger problems and a larger number of nodes, to reach a different regime for computation/communication ratio.
- Application of matrix partitioning algorithms.
- Implementation with real node failures.

Contact us

REPEAL Project
https://repeal.taa.univie.ac.at/

Carlos Pachajoa
carlos.pachajoa@univie.ac.at