Scalable, Robust, Fault-Tolerant Parallel QR Factorization

Camille Coti

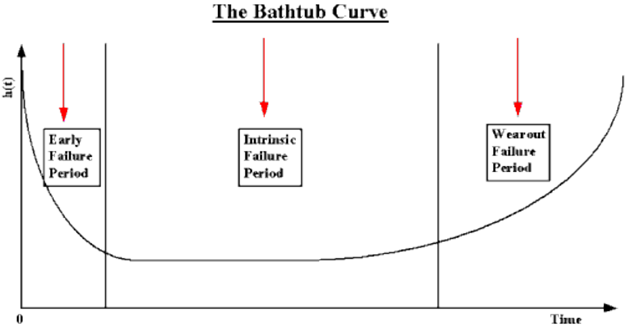
LIPN, CNRS UMR 7030, SPC, Université Paris 13, France

DCABES
August 24th, 2016

Roadmap

Large-scale systems are volatile

Life expectancy of an electronic component: the famous bathtub curve

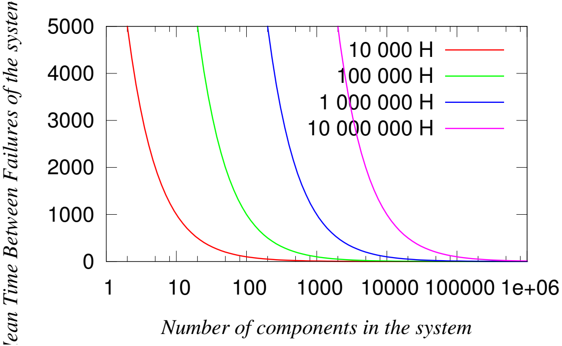


Reliability of a distributed system

Mean Time Between Failures

$$MTBF_{total} = (\sum_{i=0}^{n-1} \frac{1}{MTBF_i})^{-1} \quad (1)$$

→ The **more components** a system is made of, the **more likely** it is to have a failure.



QR Factorization

Matrix A: $A = QR$

- R is upper-triangular
- Q is orthogonal

Unicity:

- Yes, if the diagonal elements of R are positive
- No otherwise

Several algorithms:

- Dense linear algebra: algorithms based on unitary transformations.
- (Modified) Gram-Schmidt, Givens rotation, Householder projection...

QR Factorization

Panel-update approach

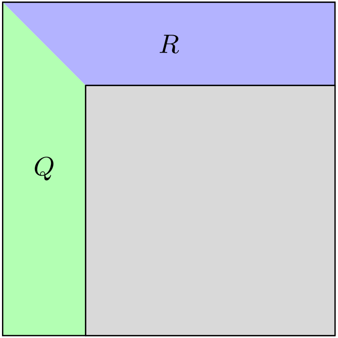
- Factorize a panel
- Update the trailing matrix
- Repeat recursively on the trailing matrix

QR Factorization

Panel-update approach

- Factorize a panel
- Update the trailing matrix
- Repeat recursively on the trailing matrix

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$
$$= Q_1 \begin{pmatrix} R_{11} & R_{12} \\ 0 & A_{22}^1 \end{pmatrix}$$

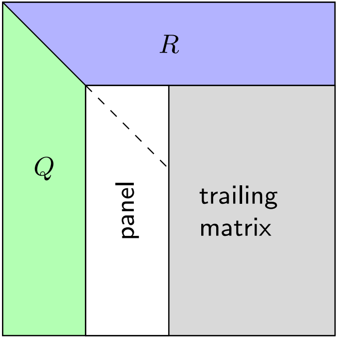


QR Factorization

Panel-update approach

- Factorize a panel
- Update the trailing matrix
- Repeat recursively on the trailing matrix

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$
$$= Q_1 \begin{pmatrix} R_{11} & R_{12} \\ 0 & A^1_{22} \end{pmatrix}$$



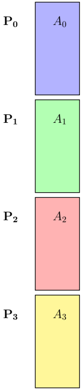
1. Introduction
   - Failures
   - QR Factorization
2. Communication-avoiding QR
   - Panel factorization
   - Trailing matrix update
3. Fault-tolerant QR
   - Panel factorization
   - Trailing matrix update
   - Resilience to soft errors
4. Conclusion

Panel factorization

QR factorization of a panel: particular shape

- $M >> N$: the matrix is **tall-and-skinny**
- Specific algorithm: **TSQR**

$$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} = Q_1 \begin{pmatrix} R_{11} \\ 0 \end{pmatrix}$$

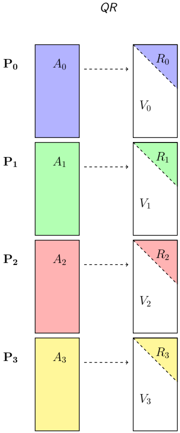


Panel factorization

QR factorization of a panel: particular shape

- $M >> N$: the matrix is **tall-and-skinny**
- Specific algorithm: **TSQR**

$$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} = Q_1 \begin{pmatrix} R_{11} \\ 0 \end{pmatrix}$$

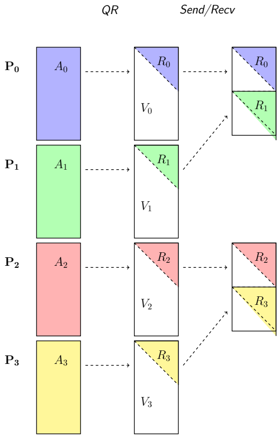


Panel factorization

QR factorization of a panel: particular shape

- $M >> N$: the matrix is **tall-and-skinny**
- Specific algorithm: **TSQR**

$$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} = Q_1 \begin{pmatrix} R_{11} \\ 0 \end{pmatrix}$$

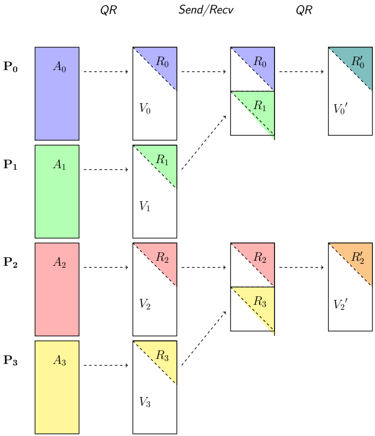


Panel factorization

QR factorization of a panel: particular shape

- $M >> N$: the matrix is **tall-and-skinny**
- Specific algorithm: **TSQR**

$$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} = Q_1 \begin{pmatrix} R_{11} \\ 0 \end{pmatrix}$$

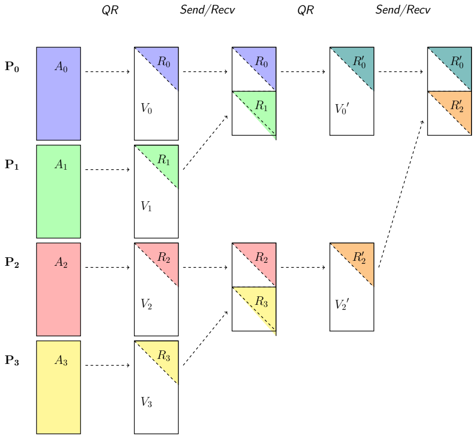


Panel factorization

QR factorization of a panel: particular shape

- $M >> N$: the matrix is **tall-and-skinny**
- Specific algorithm: **TSQR**

$$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} = Q_1 \begin{pmatrix} R_{11} \\ 0 \end{pmatrix}$$

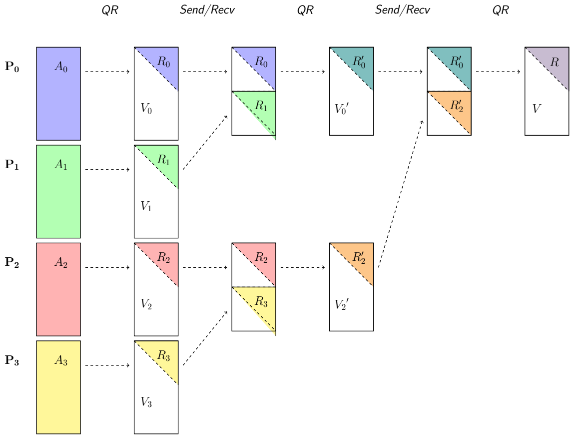


Panel factorization

QR factorization of a panel: particular shape

- $M >> N$: the matrix is **tall-and-skinny**
- Specific algorithm: **TSQR**

$$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} = Q_1 \begin{pmatrix} R_{11} \\ 0 \end{pmatrix}$$



Trailing matrix update

After the panel factorization:

- Compute the **compact representation** of Q_1: $Q_1 = I - Y_1 T_1 Y_1^T$
- Use the Y_1 and T_1 matrices to update the trailing matrix:

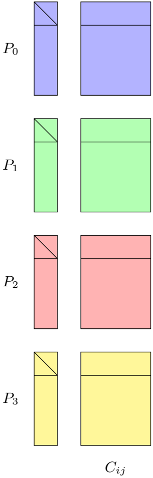
$$(I - Y_1 T_1 Y_1^T) \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix} = \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix} - Y_1 (T_1^T (Y_1^T \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix})) = \begin{pmatrix} R_{12} \\ A_{22}^1 \end{pmatrix}$$

- R_{12} : upper part of the R
- A_{22}^1 : new trailing matrix.

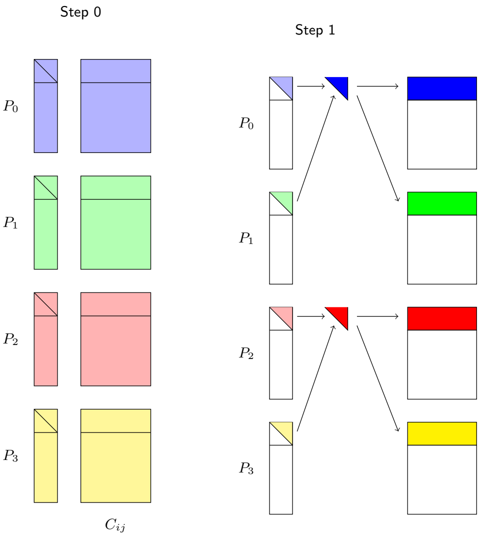
$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = Q_1 \begin{pmatrix} R_{11} & R_{12} \\ 0 & A_{22}^1 \end{pmatrix}$$

Trailing matrix update

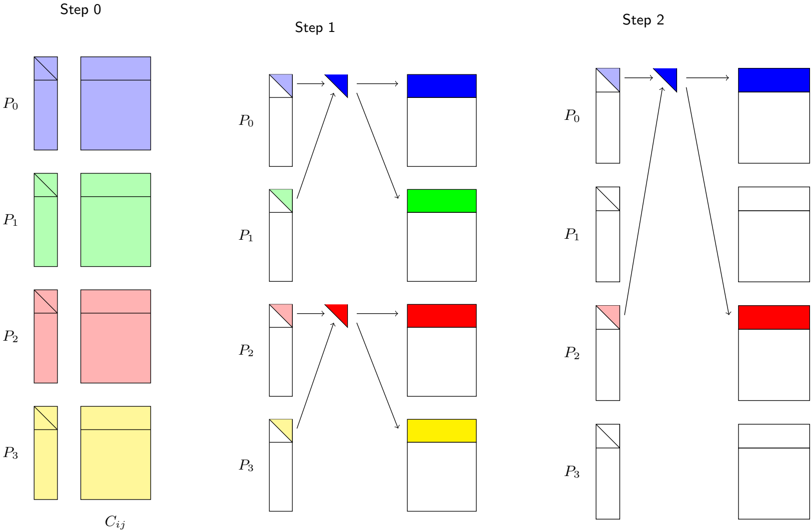
Step 0



Trailing matrix update



Trailing matrix update



1. Introduction
   - Failures
   - QR Factorization

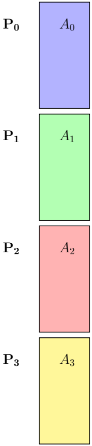
2. Communication-avoiding QR
   - Panel factorization
   - Trailing matrix update

3. Fault-tolerant QR
   - Panel factorization
   - Trailing matrix update
   - Resilience to soft errors

4. Conclusion

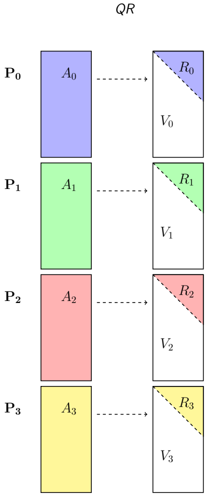
Panel factorization

Let's look at TSQR in details



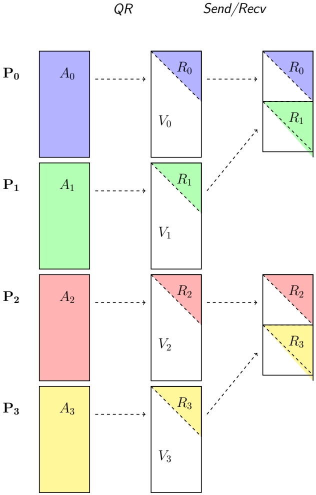
Panel factorization

Let's look at TSQR in details



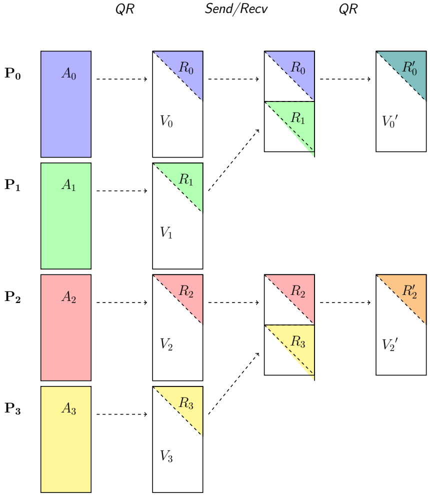
Panel factorization

Let's look at TSQR in details



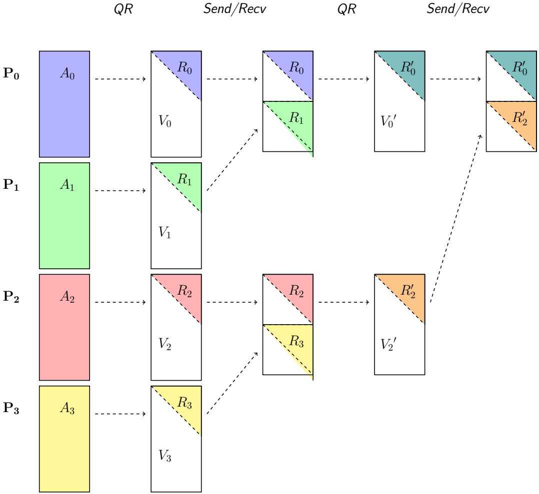
Panel factorization

Let's look at TSQR in details



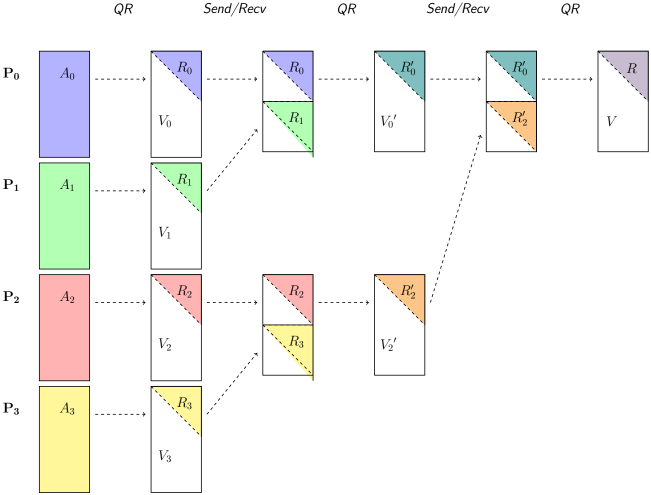
Panel factorization

Let's look at TSQR in details



Panel factorization

Let's look at TSQR in details

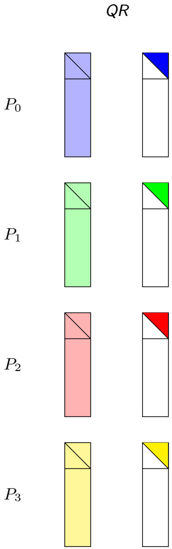


Panel factorization

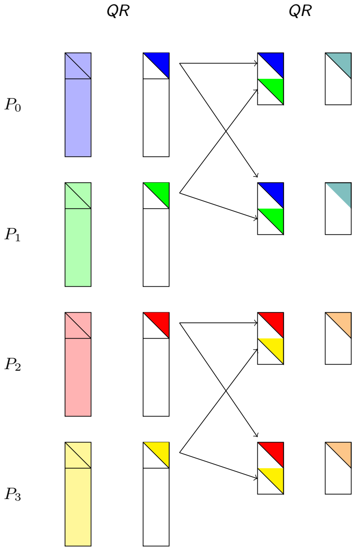
- P_0 works beginning $\rightarrow$ end
- P_2 works during the first two steps, then stops
- P_1 and P_3 work during the first step, then stops

Let's put these lazy dudes to work!

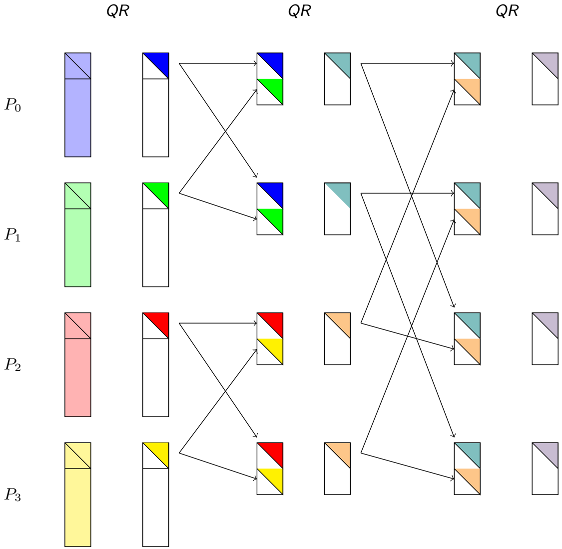
Fault-tolerant TSQR



Fault-tolerant TSQR



Fault-tolerant TSQR



Trailing matrix update

Update of the trailing matrix

- Triggered by the **tree of the panel factorization**
- Uses the intermediate Y and T matrices
- **Pairwise operation** $\rightarrow$ tree

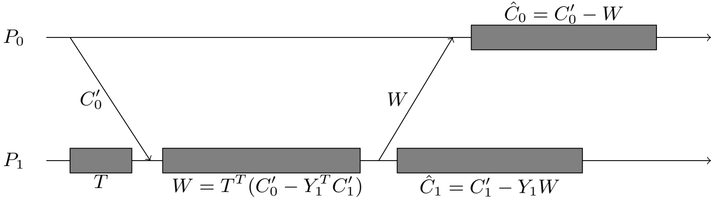
$$A = Q \begin{pmatrix} R & \hat{C}'_0 \\ & \hat{C}'_1 \end{pmatrix}$$

Decompose the blocks of the trailing matrix:

$$C_i = \begin{pmatrix} C'_i \\ C''_i \end{pmatrix} = \begin{pmatrix} C_i[: N-1] \\ C_i[N :] \end{pmatrix}$$

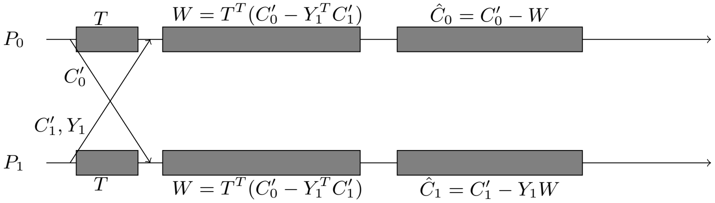
Compute the update :

$$\begin{pmatrix} R_0 & C'_0 \\ R_1 & C'_1 \end{pmatrix} = \begin{pmatrix} QR & C'_0 \\ & C'_1 \end{pmatrix}$$



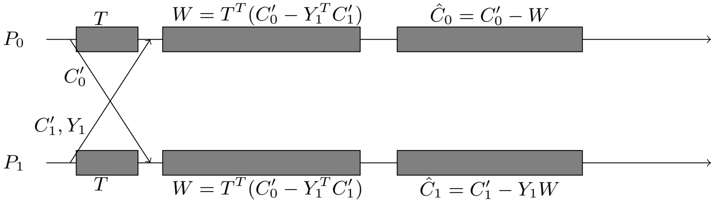
Fault-tolerant update

Change the order of the operations



Fault-tolerant update

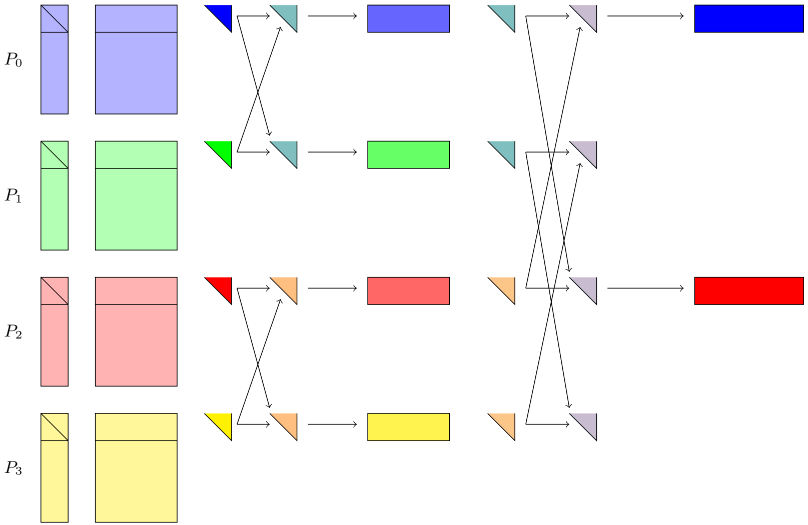
Change the order of the operations



If a process fails

- Both process have the W matrix
- Computation recovered from the W matrix and the initial submatrix

Communication scheme



Soft errors

Soft errors :

- Bit flips causing numerical errors
- Caused by radiations, cosmic rays, various electric disturbances...

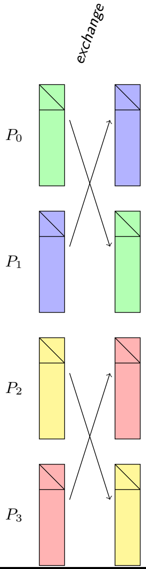
Approaches to tackle soft errors:

- Triple modular redundancy
- ECC memory
- Multiple executions
- ...
- At several levels: hardware, system, application

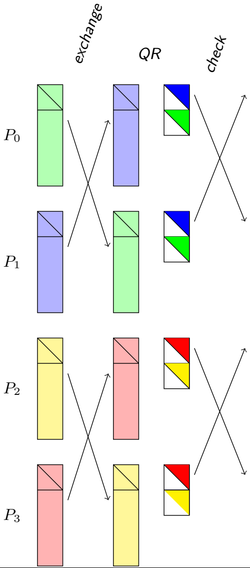
Idea here:

- **Take advantage of partial redundancy** to detect errors
- Recompute only the erroneous step

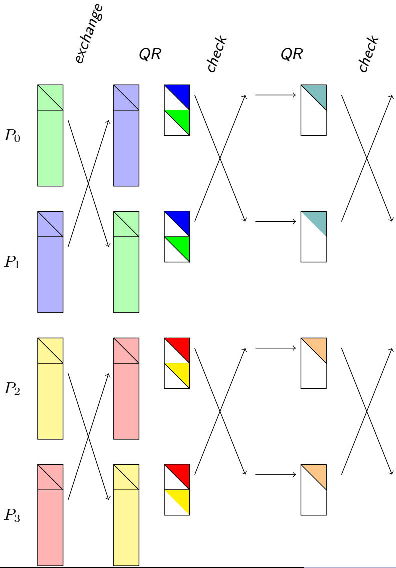
Soft errors



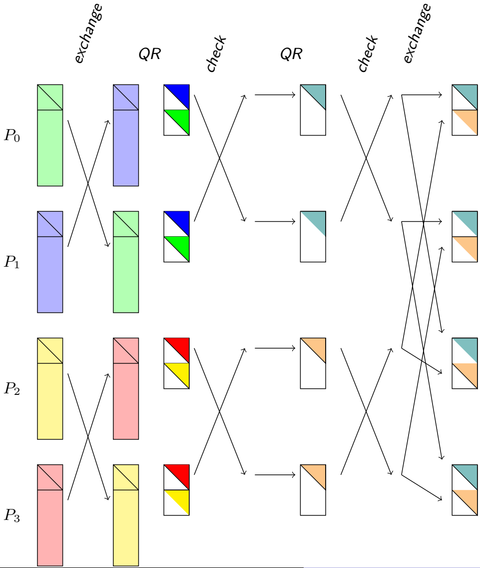
Soft errors



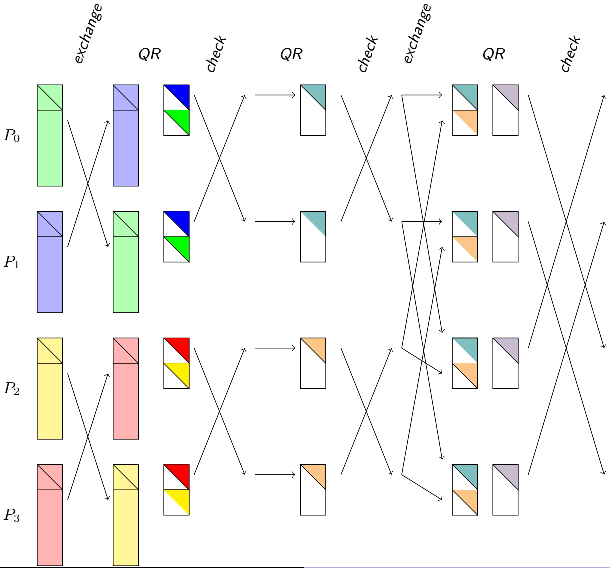
Soft errors



Soft errors



Soft errors



Conclusion

Algorithm-based fault-tolerance

- Failure recovery is handled by the application
- Behavior upon failures: **defined in the algorithm**
- Goal:
  - Minimal overhead during failure-free executions
  - As few recomputations as possible after failures

Fault-tolerant algorithms for QR factorization

- Use **intrinsic redundancy** or idle processes
- **Operation reordering** based on algebraic properties

Small modifications in the critical path

- No additional computations
- Data exchange instead of send/recv
- Partial results already ready for recovery

Appendix: performance