UPPSALA
UNIVERSITET

Recursive inverse factorization

Anton Artemov

Division of Scientific Computing, Uppsala University

anton.artemov@it.uu.se

06.09.2017

Context of work

UPPSALA UNIVERSITET

Research group on large scale electronic structure computations at TDB, UU:

- **Ergo** - a quantum chemistry program for large-scale self-consistent field calculations. `http://ergoscf.org`
- Parallel programming models
- **Chunks and Tasks** model and library `http://chunks-and-tasks.org`

E. Rubensson, E. Rudberg, A. Kruchinina, A. Artemov.

Introduction

UPPSALA UNIVERSITET

- Electron density matrix D for a given Fock or Kohn–Sham martix F:

$$D = \sum_{i=1}^{nocc} c_i c_i^T, \tag{1}$$

where c_i are eigenvector solutions of the generalized eigenvalue problem:

$$Fc_i = \lambda_i S c_i, \;\; i = 1, \ldots, n, \tag{2}$$

where $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{nocc} \leq \lambda_{nocc+1} \leq \ldots \leq \lambda_n$, $nocc$ is the number of occupied electron orbitals and S is the basis set overlap matrix.

- Standard way to compute D is via complete solution of eigenvalue problem.

Introduction

UPPSALA UNIVERSITET

- Simple and efficient method is to see the problem as a matrix function

$$D = \Theta(\mu I - F), \tag{3}$$

where Θ is Heaviside function, $\mu \in]\lambda_{nocc}, \lambda_{nocc+1}[$ and to use polynomial expansion of F.

- Sparse matrix data structures give $O(n)$ complexity;
- expansion can be performed directly in F;
- more efficient to perform congruence transformation:

$$S \to Z^T S Z = I, \; F \to Z^T F Z = \hat{F}. \tag{4}$$

and transform eigenvalue problem (2) into standard form:

$$\hat{F} y = \lambda y, \; Z y = x, \; S^{-1} = Z Z^T. \tag{5}$$

Motivation

UPPSALA UNIVERSITET

- Common techniques for inverse factorization:
  - Löwdin
  - Inverse Cholesky

  provide decay of matrix elements.
- Generally their complexity is $O(n^3)$.
- Alternatively: iterative refinement, but sufficiently accurate starting guess is necessary.
- Main idea - combine iterative refinement with a binary principal submatrix decomposition as starting guess.

Iterative refinement

UPPSALA UNIVERSITET

- Non-local refinement:

$$Z_{i+1} = Z_i \sum_{k=0}^{m} b_k \delta_i^k, \;\; b_k = \frac{2k-1}{2k}, \;\; b_0 = 1, \;\; \delta_i = I - Z_i^* S Z_i. \tag{6}$$

- Local refinement:

$$\begin{cases} \delta_{i+1} = \delta_i - Z_{i+1}^* S (Z_{i+1} - Z_i) - (Z_{i+1} - Z_i)^* S Z_i, \\ Z_{i+1} = Z_i \sum\limits_{k=0}^{m} b_k \delta_i^k. \end{cases} \tag{7}$$

Recursive Inverse Factorization

UPPSALA UNIVERSITET

Algorithm REC-INV-FACT

Input: Hermitian positive definite matrix S

Output: Z

1: **if** lowest level **then**
2: Factorize $S^{-1} = ZZ^*$ and **return** Z
3: **end if**
4: Matrix partition $S = \begin{bmatrix} A & B \\ B^* & C \end{bmatrix}$
5: $Z_A = \text{REC-INV-FACT}(A)$, $Z_C = \text{REC-INV-FACT}(C)$
6: $Z_0 = \begin{bmatrix} Z_A & 0 \\ 0 & Z_C \end{bmatrix}$
7: $\delta_0 = I - Z_0^* S Z_0$
8: **repeat**
9: $Z_{i+1} = Z_i \sum_{k=0}^{m} b_k \delta_i^k$
10: $\delta_{i+1} = I - Z_{i+1}^* S Z_{i+1}$
11: **until** $\|\delta_{i+1}\| > \|\delta_i\|^{m+1}$
12: **return** Z_{i+1}

Dynamical hierarchical algorithms

- Dense matrix-matrix-multiplication is good for parallelization.
- Here, nonzero pattern is not known in advance and may change dynamically.
- This makes parallelization for distributed memory difficult.

Conventional parallelization (MPI)

Application programmer responsibilities:

Conventional parallelization (MPI)
Distribute work
Distribute data
Send and receive messages

MPI vs Chunks and Tasks

Application programmer responsibilities:

Conventional parallelization (MPI)	Chunks and Tasks
Distribute work	Divide work into smaller parts
Distribute data	Divide data into smaller pieces
Send and receive messages	Register “tasks” and “chunks”

Chunks and Tasks programming model

- Developed for dynamic hierarchical algorithms
- Abstractions for both data and work (chunks and tasks)
- No explicit communication calls in user code
- Determinism, freedom from race conditions and deadlocks
- Feasible to implement efficient backends
- Fail safety: local recovery possible

Chunks and Tasks programming model

UPPSALA UNIVERSITET

- Developed for dynamic hierarchical algorithms
- Abstractions for both data and work (chunks and tasks)
- No explicit communication calls in user code
- Determinism, freedom from race conditions and deadlocks
- Feasible to implement efficient backends
- Fail safety: local recovery possible

Main concepts:

- **Chunk** is defined by data members, serialization functions, may contain identifiers to other chunks giving rise to chunk hierarchies.
- **Task** is defined by input chunk types, output chunk type, work to be performed.

Related programming models

UPPSALA UNIVERSITET

Computation driven by registration of tasks

Like Cilk, StarPU, SuperGlue, XKaapi but unlike Charm++ (where the computation is driven by messages)

$\Longrightarrow$ No message passing

Recursive nesting of tasks allowed

Like Cilk, SuperGlue, XKaapi but unlike StarPU

$\Longrightarrow$ Scalable, good for hierarchic dynamic algorithms

Abstractions for both work and data

Previous task-based approaches mostly focused on shared memory. Distributed memory: either user provides data distribution or all data is managed by a "master" node

$\Longrightarrow$ User does not have to provide data distribution

Related programming models

UPPSALA UNIVERSITET

Identifiers to chunks provided by the library

Unlike Linda and Concurrent Collections

$\Longrightarrow$ Makes it feasible to make data available efficiently

$\Longrightarrow$ Leads to restrictions in how data can be accessed

Chunks are readonly once they are registered

Unlike Linda but similar to Concurrent Collections

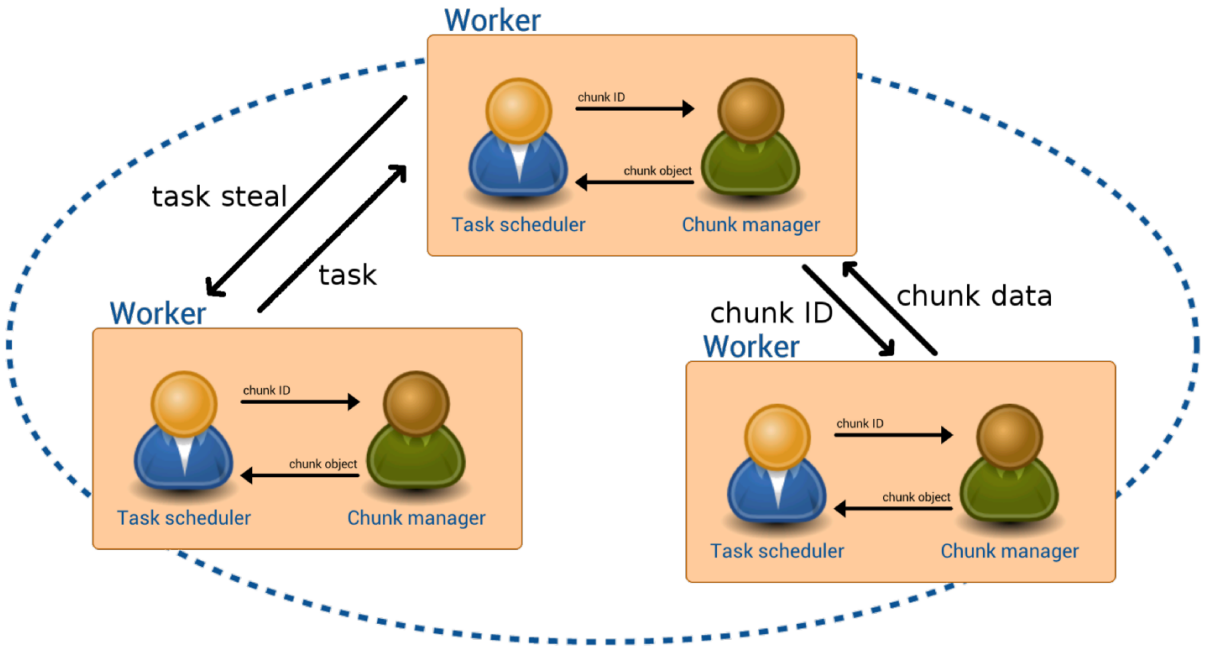
$\Longrightarrow$ Chunk cache coherence not an issue

$\Longrightarrow$ Determinism (easier for the user to understand her code)

CHT-MPI library

UPPSALA UNIVERSITET

- Publicly available since May 2014 (modified BSD license)
- Task scheduler based on work stealing
- Scalable: No "master node" managing all work or data

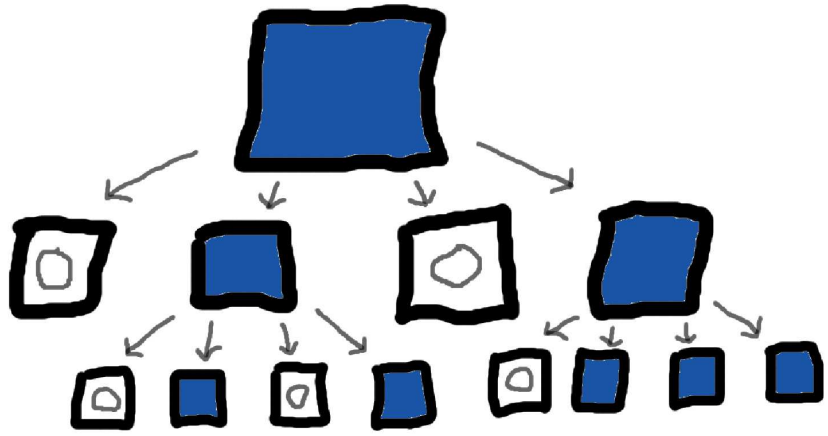


Webpage: `www.chunks-and-tasks.org`
Ref:[Parallel Computing, **40**: 328-343 (2014)]

Chunks and Tasks matrix library

UPPSALA UNIVERSITET

- Sparse quad-tree representation:



- Different leaf matrix representations can be used.
- Sparsity is dynamically exploited at all levels by skipping operations on zero submatrices.

Result: close to linear scaling with system size for matrix-matrix multiplication for block-sparse leaf level matrices [Parallel Computing, **57**: 87-106 (2016)].

Experimental results

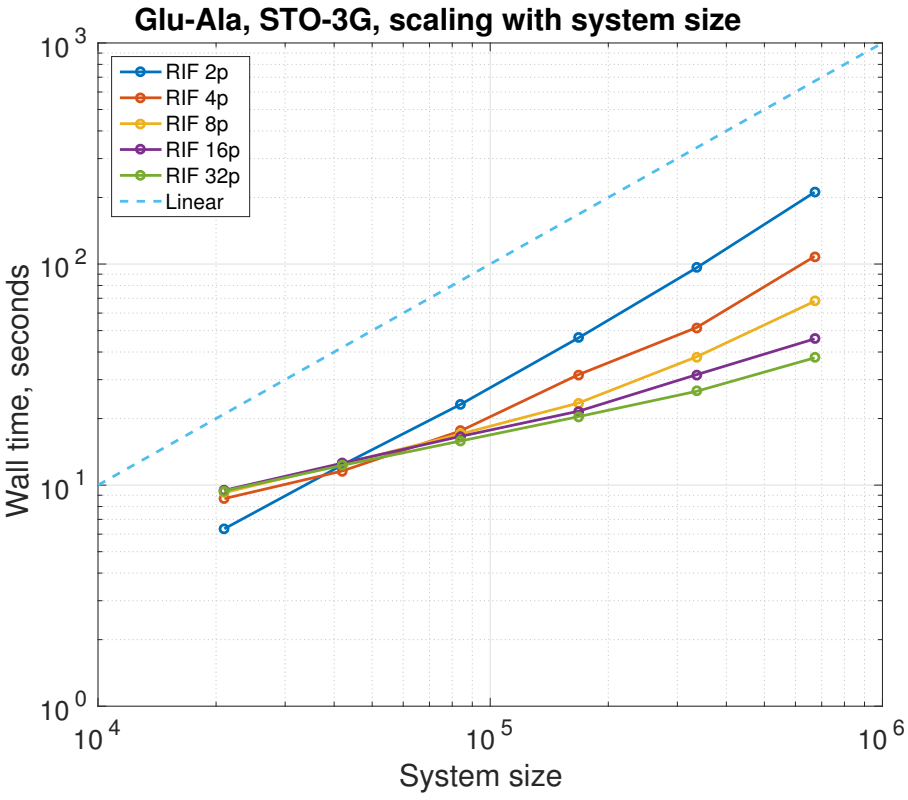
UPPSALA UNIVERSITET

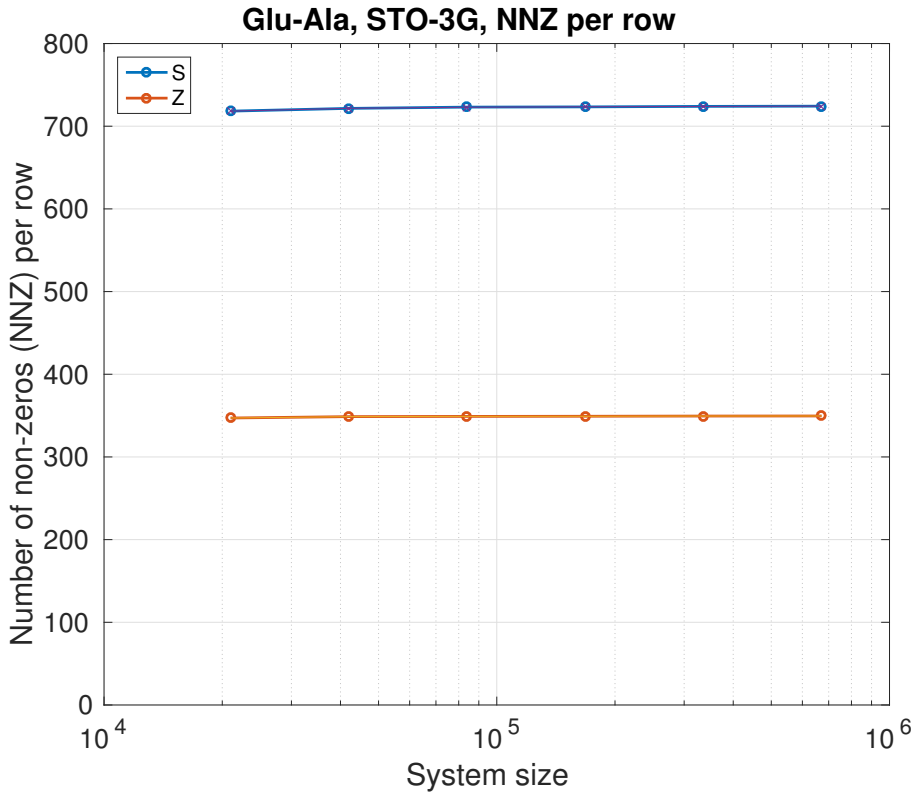
Experimental setup:

- Two kinds of molecules used: Glu-Ala helices (quasi-linear) and three-dimensional water clusters.
- Two standard Gaussian basis sets: STO-3G and the larger 3-21G basis set.
- Parallelization done with CHT-MPI.
- Recursive inverse Cholesky with truncation is employed when reaching certain size.
- Computations were performed on Beskow, a Cray XC40 system with Intel Xeon E5-2698v3 cores.
- Resources provided by Swedish National Infrastructure for Computing (SNIC) through PDC Center for High Performance Computing at the KTH Royal Institute of Technology in Stockholm.

Experimental results

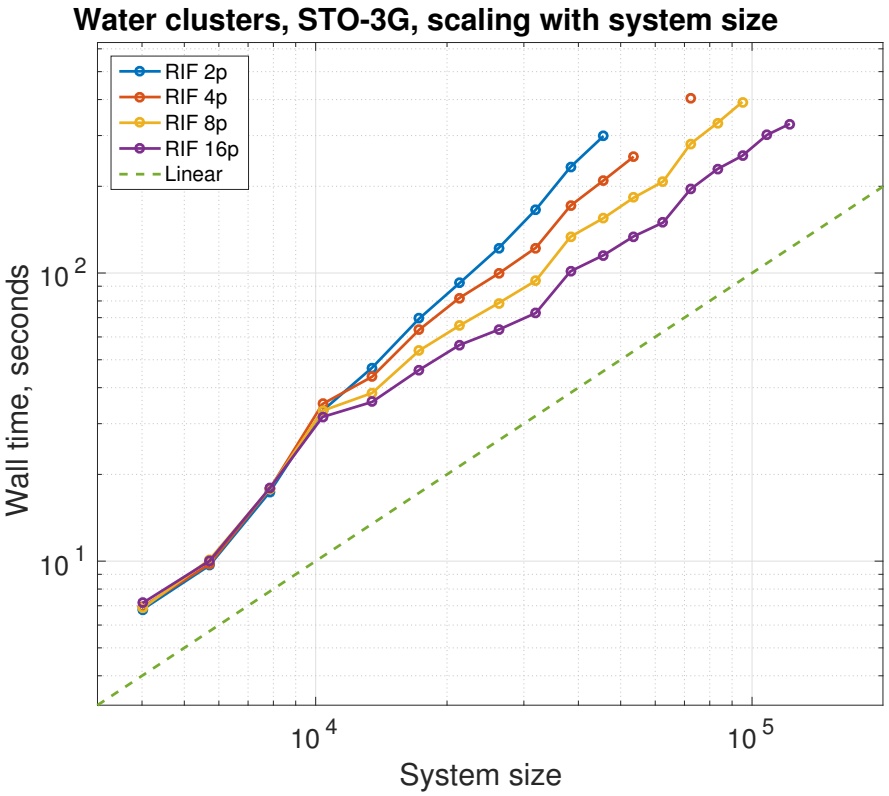
UPPSALA UNIVERSITET

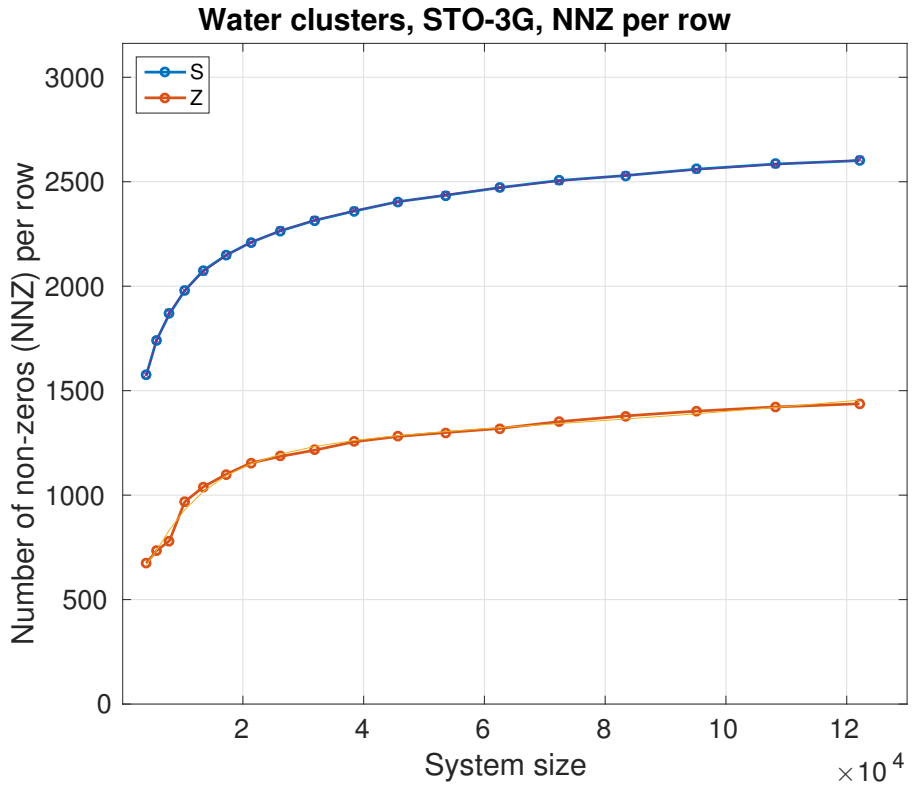




Experimental results

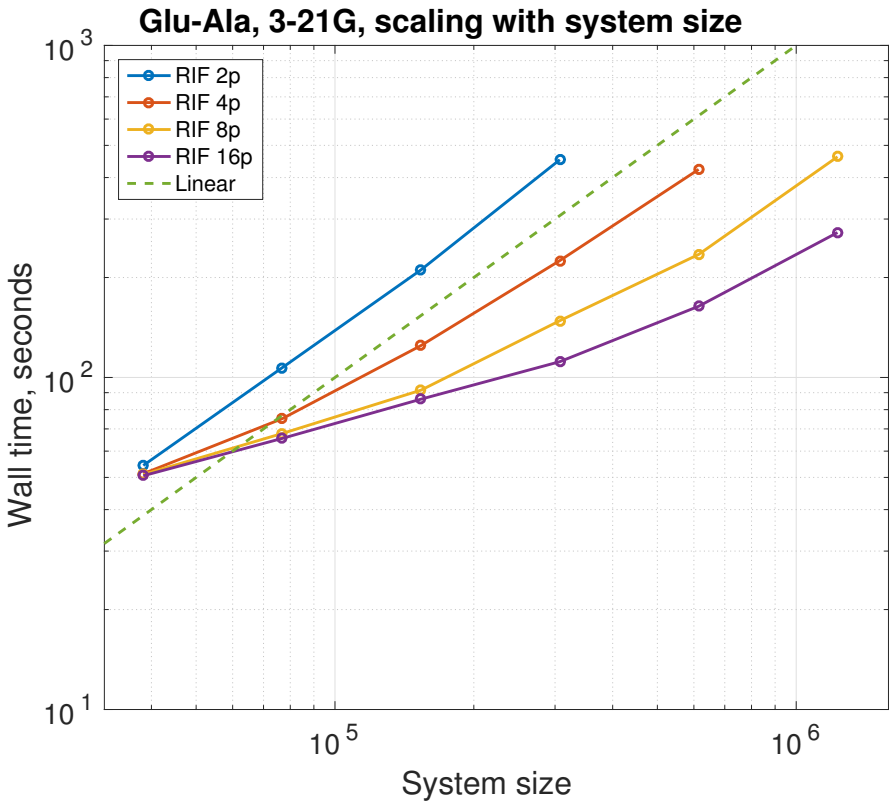
UPPSALA UNIVERSITET

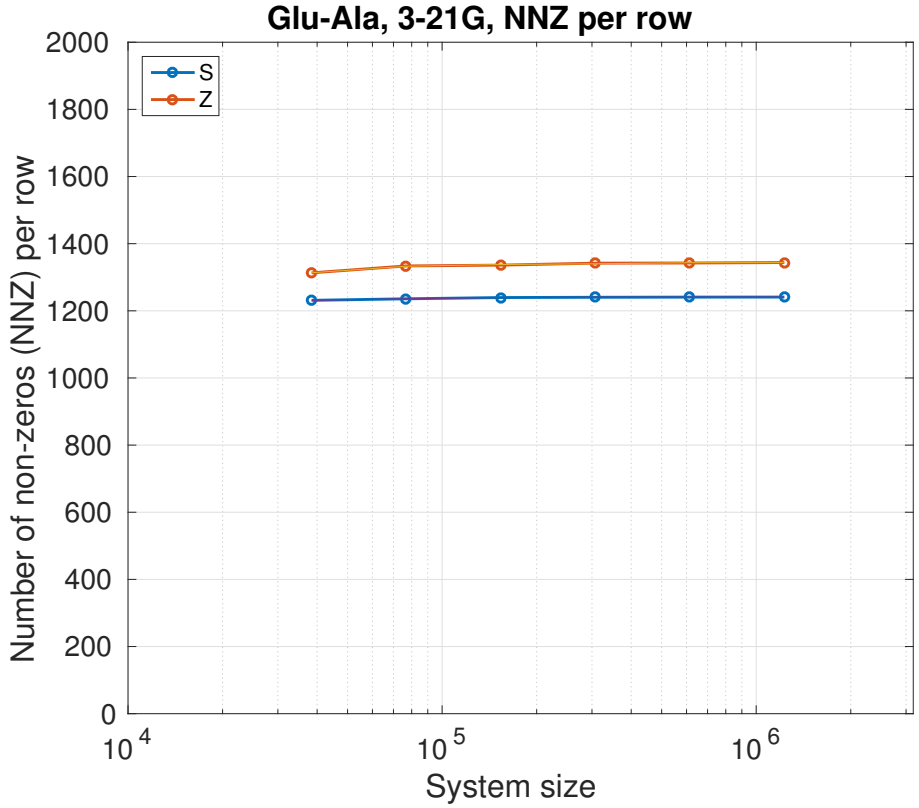




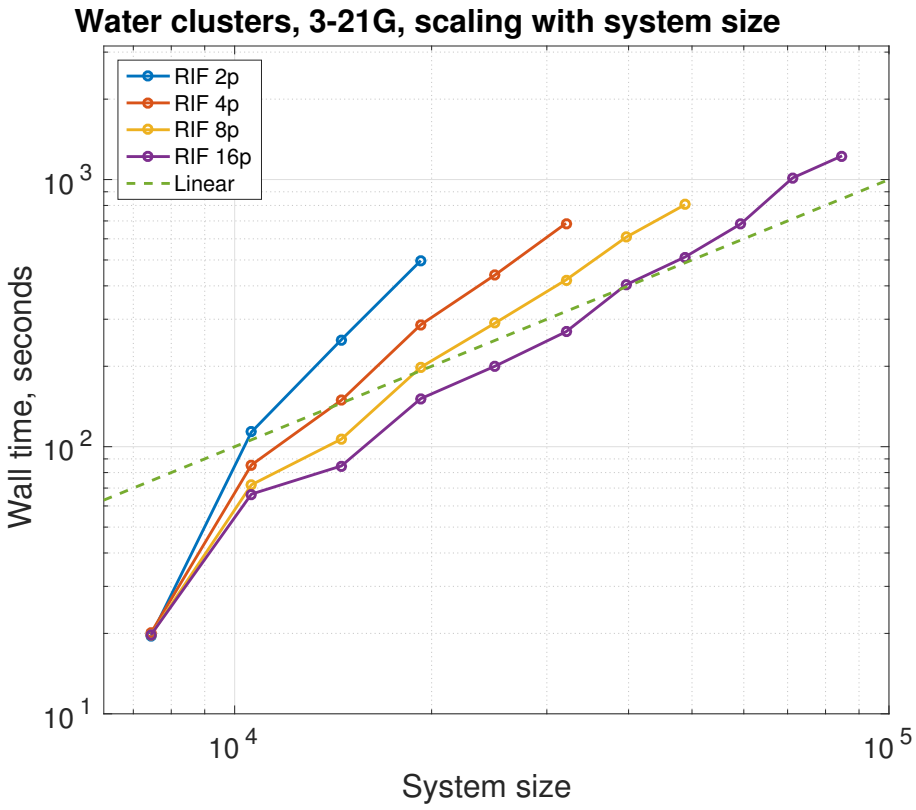
Experimental results

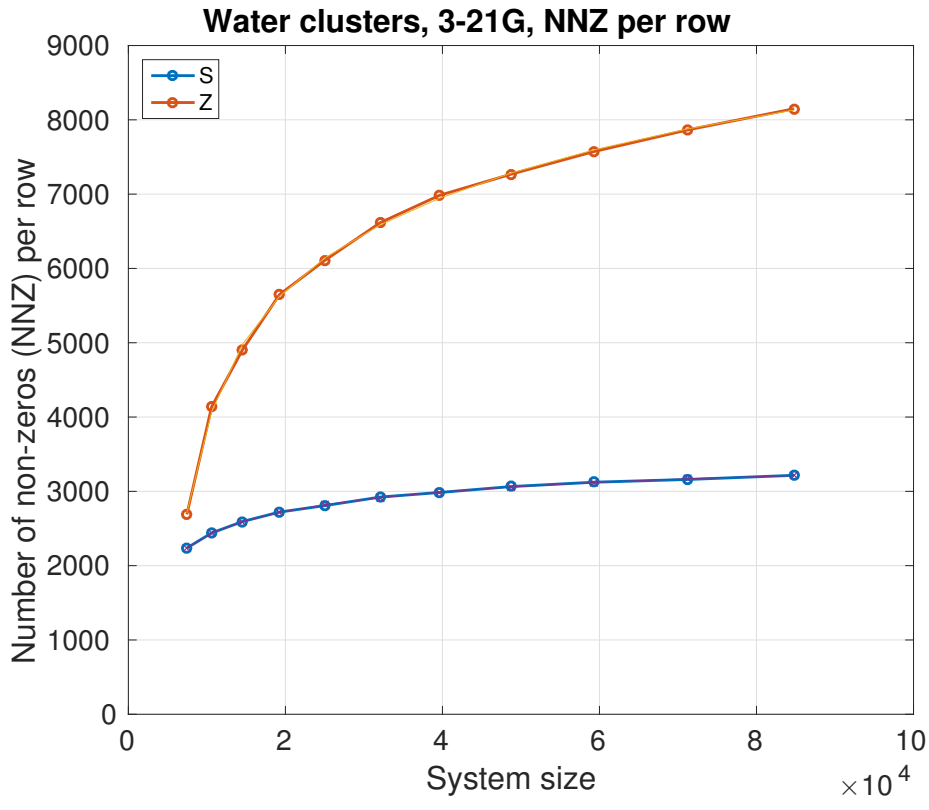
UPPSALA UNIVERSITET





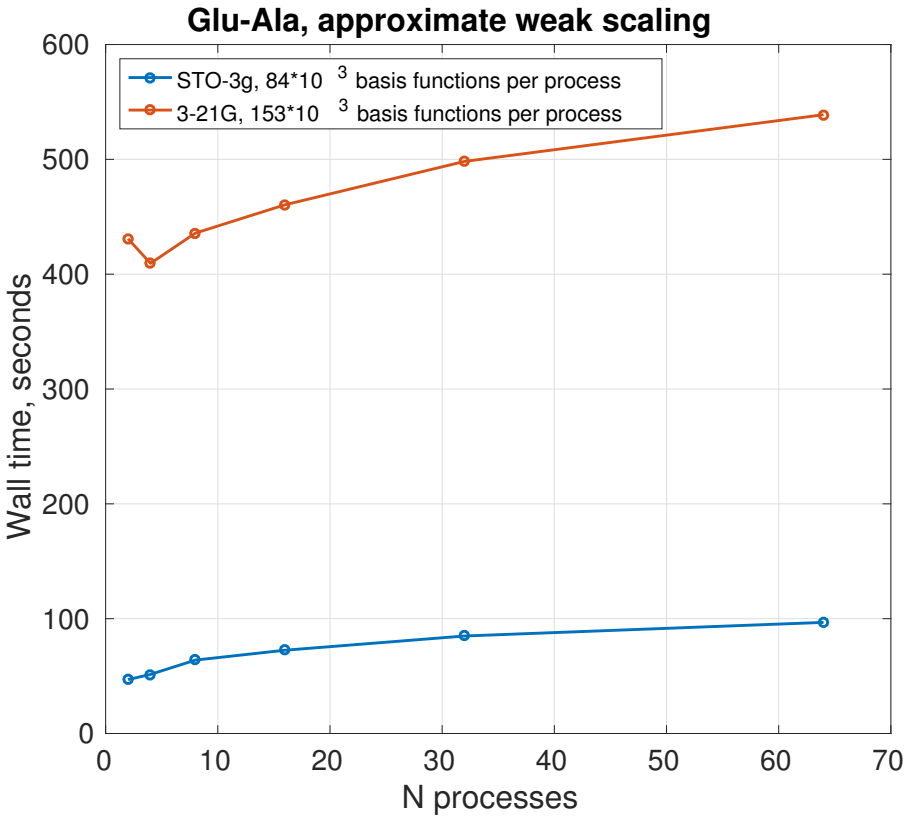
Experimental results

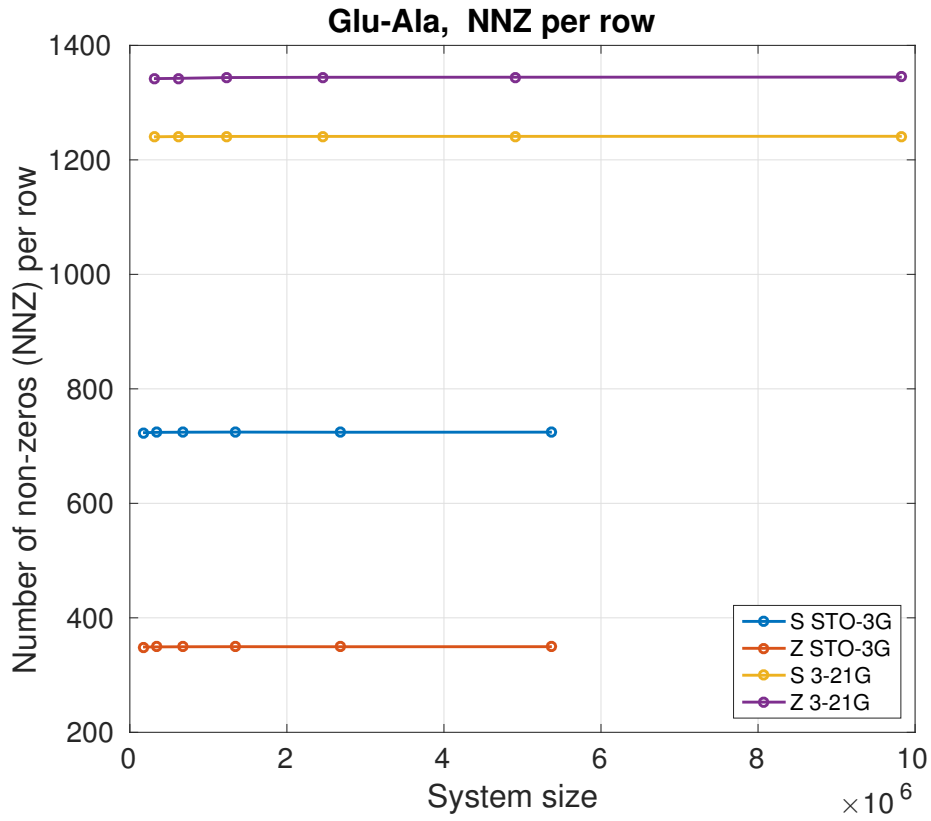




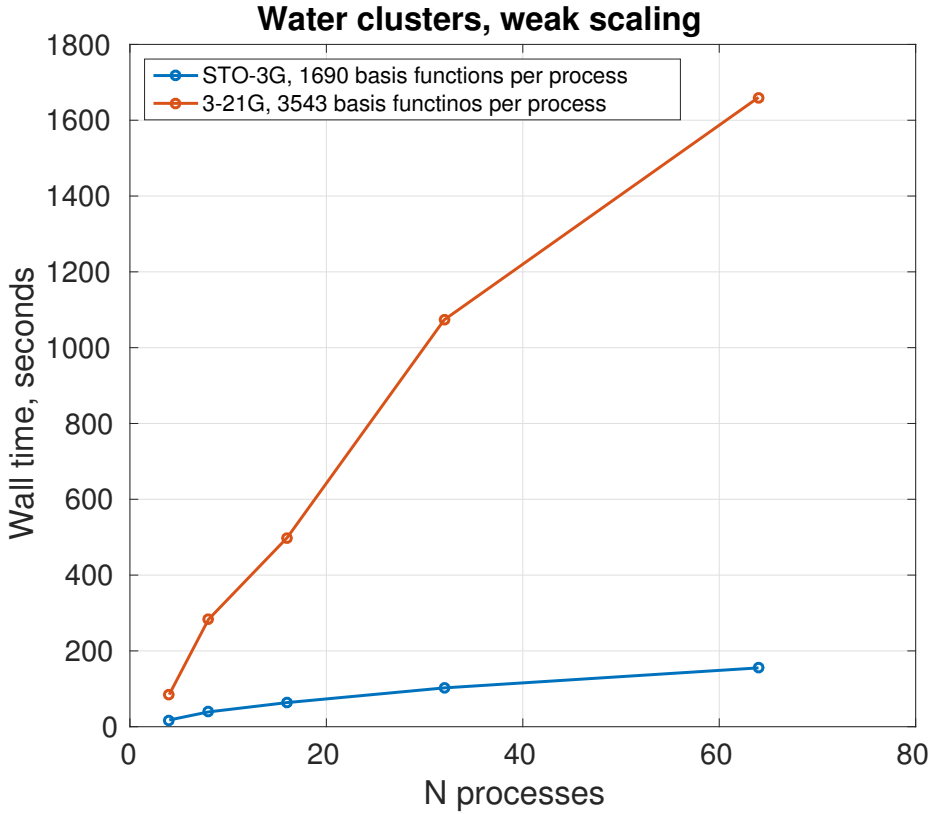
Experimental results

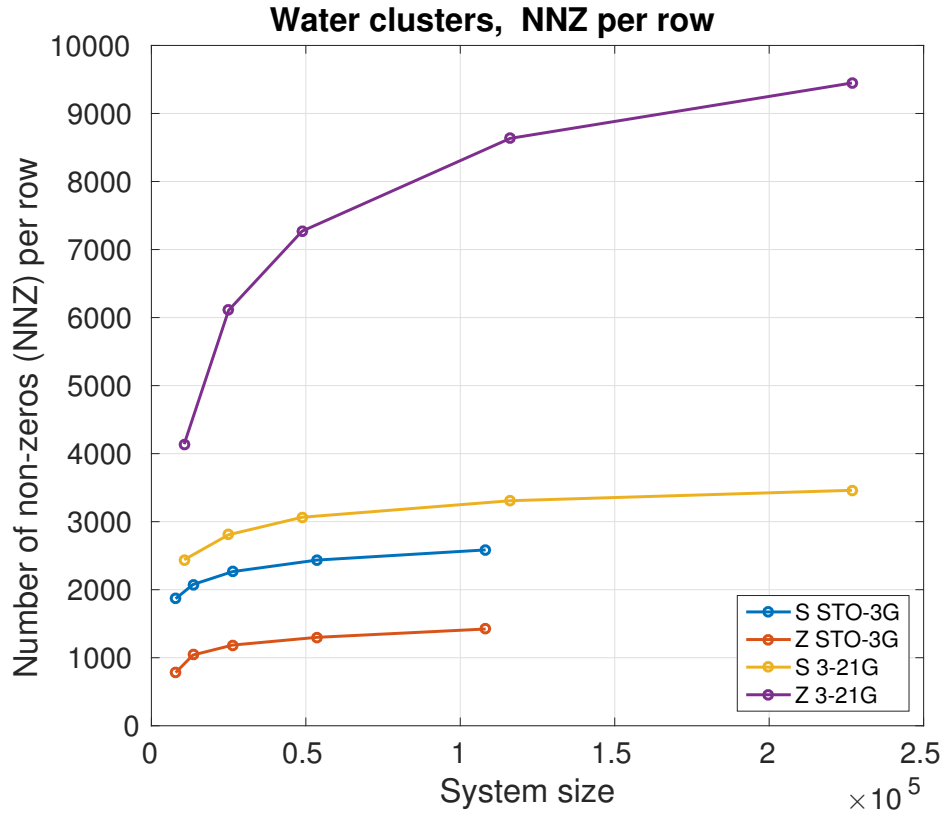
UPPSALA UNIVERSITET





Experimental results





Critical path

UPPSALA UNIVERSITET

- **Critical path** — the total amount of work required by an ideal infinite-processor execution.
- Or the longest sequence of operations, which have to be executed serially due to dependencies.
- Let $\Theta(N)$ be the critical path length (c.p.l.) for matrix recursive inverse factorization, $\xi(N)$ be c.p.l. of matrix-matrix multiplication.

Assumptions:

- Leaf level are of size $bs \Rightarrow L = \log_2(N/bs)$ levels in the hierarchy.
- Matrix sum has c.p.l. $\log_2(N)$, leaf-level routines have c.p.l. of 1.

Critical path

UPPSALA UNIVERSITET

Then

- Iterative refinement has c.p.l. $C_1 \cdot \xi(N_{cur}) + C_2 = \mu(N_{cur})$.
- Total number of tasks:

$$\sum_{i=0}^{L-1} 2^i \left(1 + \mu\left(\frac{N}{2^i}\right)\right) + 2^L. \tag{8}$$

Critical path

UPPSALA UNIVERSITET

Then

- Iterative refinement has c.p.l. $C_1 \cdot \xi(N_{cur}) + C_2 = \mu(N_{cur})$.
- Total number of tasks:

$$\sum_{i=0}^{L-1} 2^i \left(1 + \mu\left(\frac{N}{2^i}\right)\right) + 2^L. \tag{8}$$

- All recursive calls are handled in parallel:

$$\Theta(N) = \sum_{i=0}^{L-1} \left(1 + \mu\left(\frac{N}{2^i}\right)\right) + 1. \tag{9}$$

Critical path

Then

- Iterative refinement has c.p.l. $C_1 \cdot \xi(N_{cur}) + C_2 = \mu(N_{cur})$.
- Total number of tasks:

$$\sum_{i=0}^{L-1} 2^i \left(1 + \mu\left(\frac{N}{2^i}\right)\right) + 2^L. \tag{8}$$

- All recursive calls are handled in parallel:

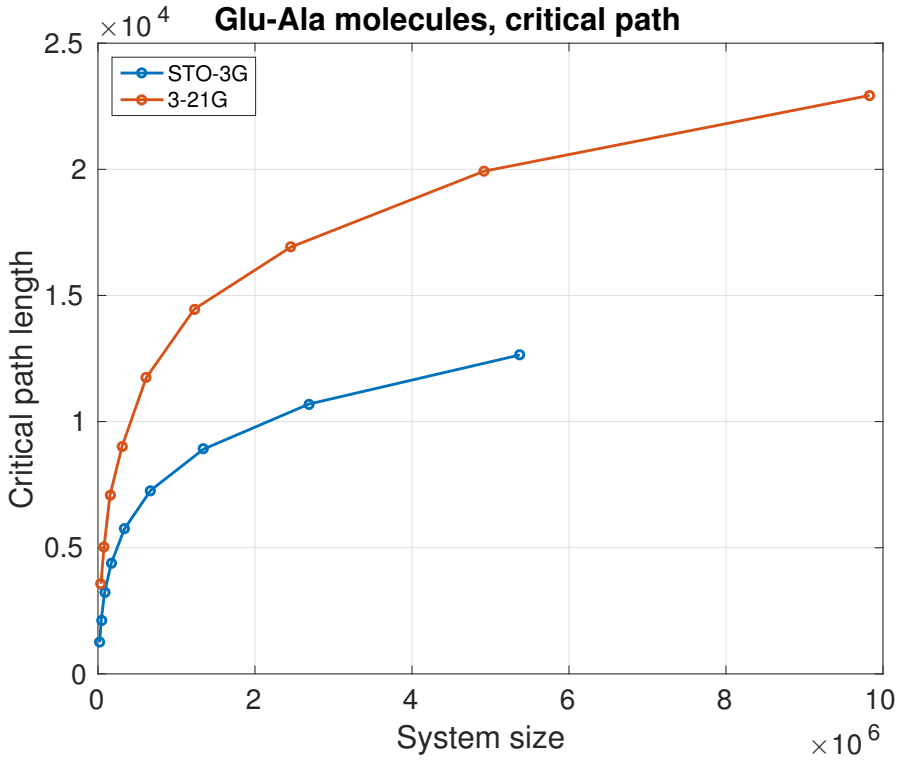
$$\Theta(N) = \sum_{i=0}^{L-1} \left(1 + \mu\left(\frac{N}{2^i}\right)\right) + 1. \tag{9}$$

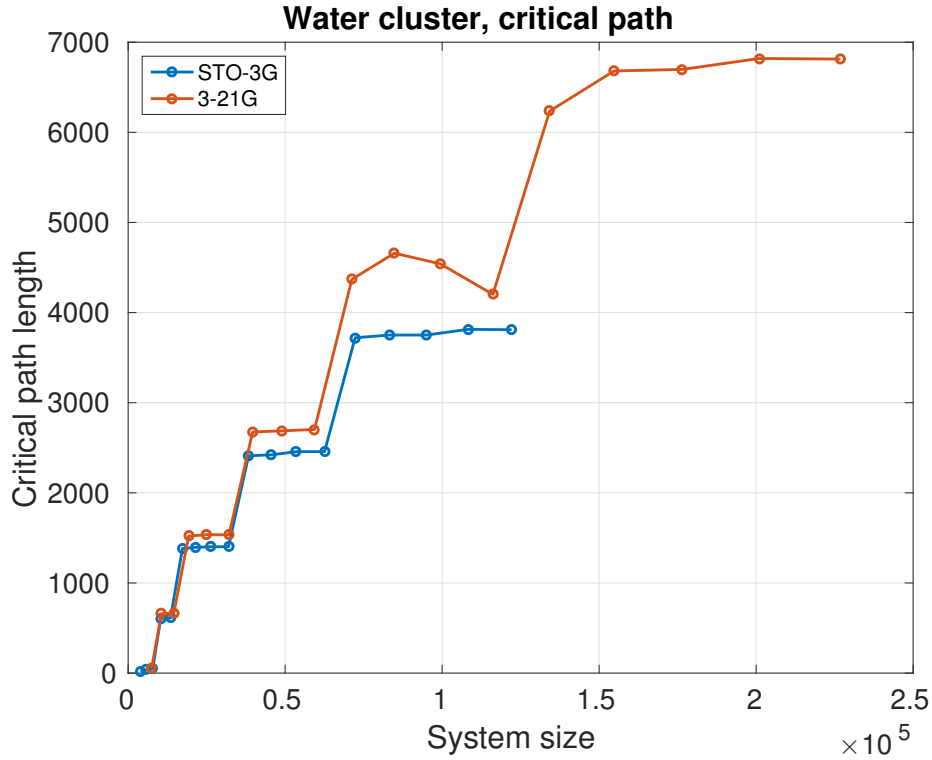
- Assuming $\xi(N_{cur}) = \log_2^2(N_{cur})$:

$$\Theta(N) \leq O(\log_2^3(N)); \tag{10}$$

Experimental results

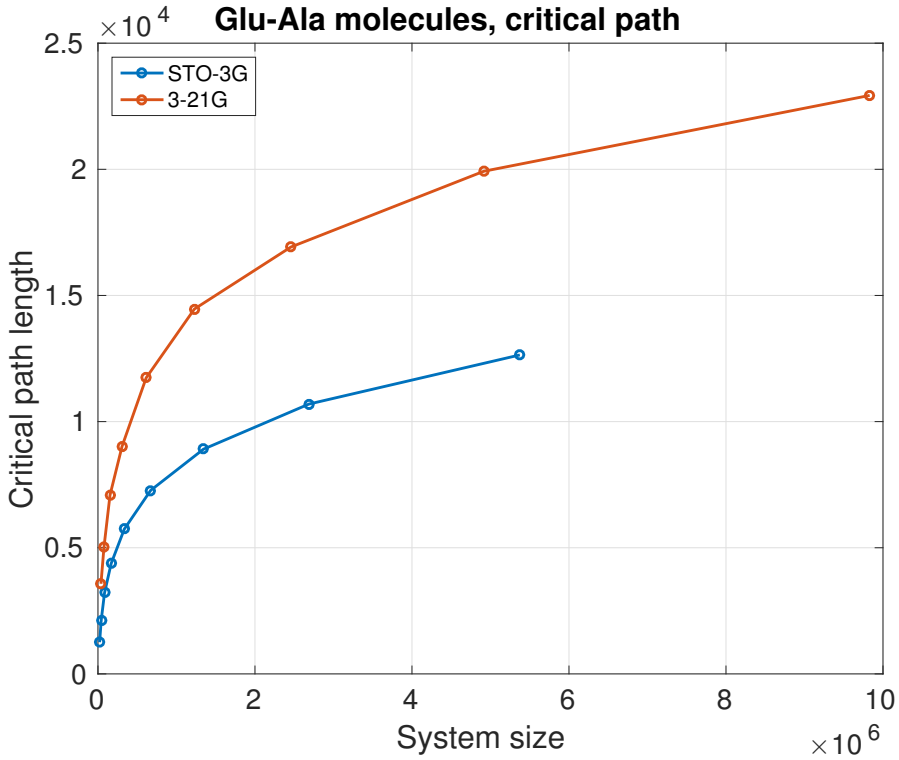
UPPSALA UNIVERSITET

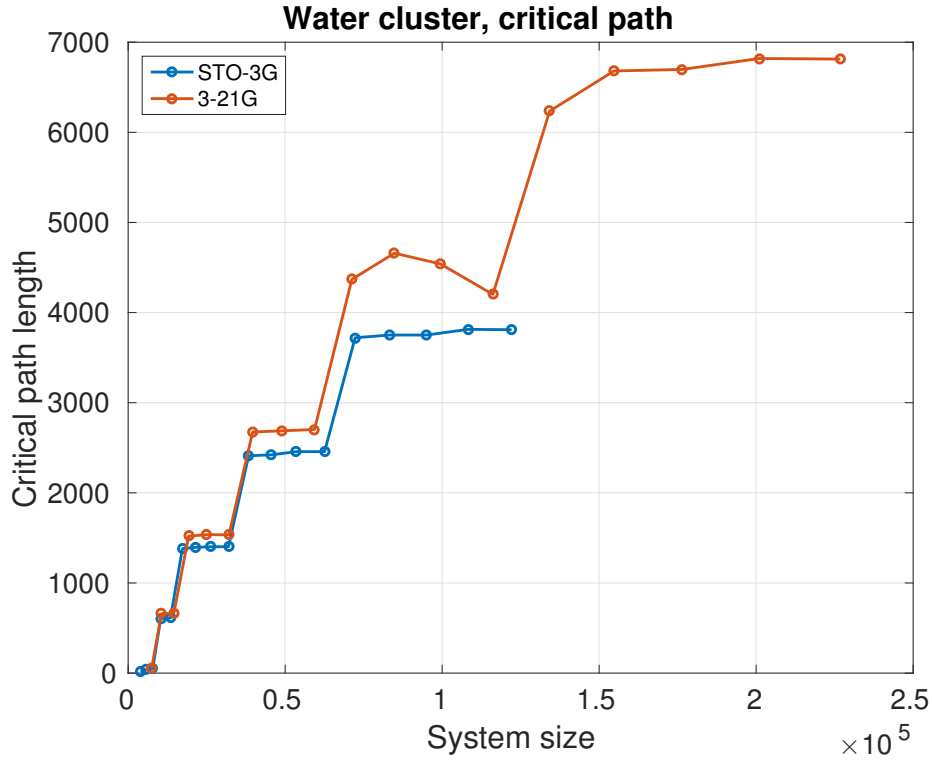




Experimental results

UPPSALA UNIVERSITET





Least-squares fitting shows that c.p.l. grows like $O(\log_2^2(N))$ rather than $O(\log_2^3(N))$

Conclusions

UPPSALA
UNIVERSITET

Conclusions

Presented algorithm

- can be built on top of more demanding algorithm used on leaf-level computations;
- scales almost linearly with system size;
- has weak scaling better than any power function;
- has critical path with length $\leq O(\log_2^3(N))$.

References

- Emanuel H. Rubensson, Elias Rudberg, Chunks and Tasks: A programming model for parallelization of dynamic algorithms, Parallel Computing, Volume 40, 2014, Pages 328-343.
- Emanuel H. Rubensson, Elias Rudberg, Locality-aware parallel block-sparse matrix-matrix multiplication using the Chunks and Tasks programming model, Parallel Computing, Volume 57, 2016, Pages 87-106.
- `www.chunks-and-tasks.org`

The End

Any Questions?