# Domain Decomposition at 1.85M MPI Processes

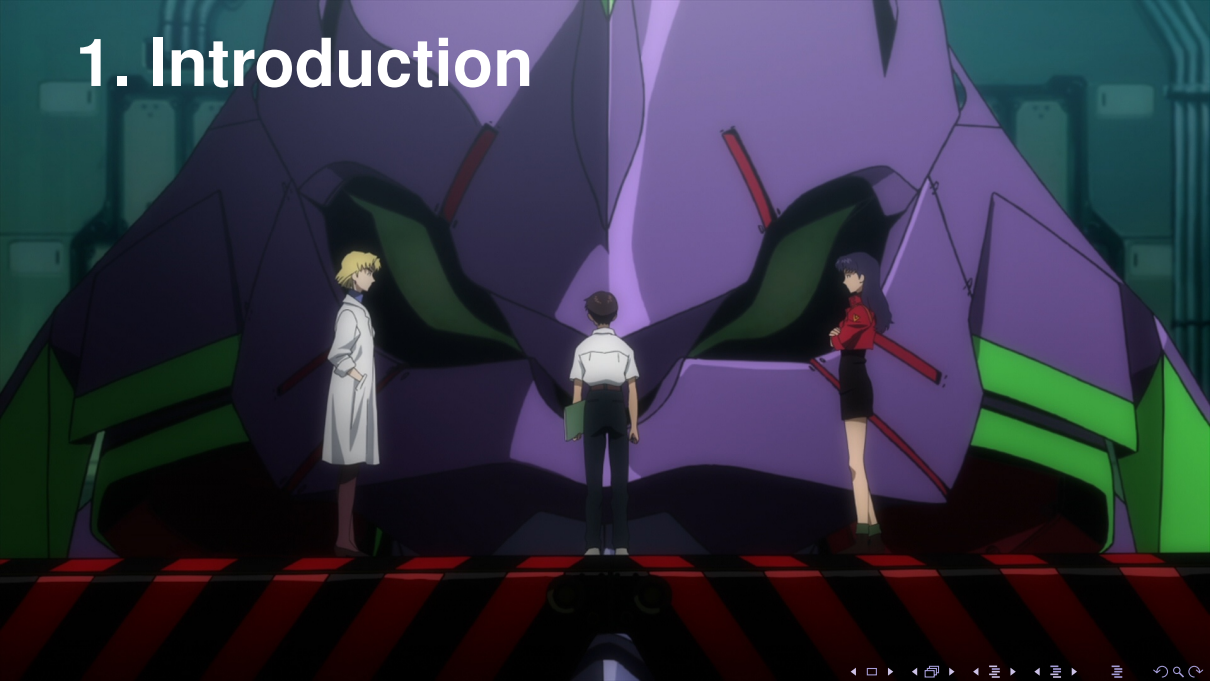
Andreas Schäfer
andreas.schaefer@fau.de

Friedrich-Alexander-Universität Erlangen-Nürnberg

Extreme Scale Programming Tools Workshop
SC13, Denver, Colorado
2013.11.18

# Outline

# 1. Introduction



# What is Domain Decomposition?

- partition of simulation graph
- one domain per rank
- goals
  1. minimize communication (total volume vs. max individual)
  2. low overhead
  3. equalize load
- challenges
  1. decomposition must match simulation model
  2. analytic evaluation vs. real hardware
  3. decomposition technique tied to parallelization

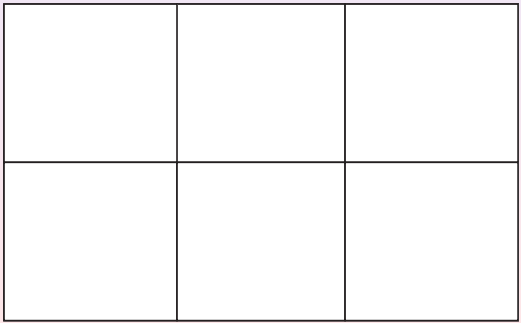
# What is Domain Decomposition?

- partition of simulation graph
- one domain per rank
- goals
  1. minimize communication (total volume vs. max individual)
  2. low overhead
  3. equalize load
- challenges
  1. decomposition must match simulation model
  2. analytic evaluation vs. real hardware
  3. decomposition technique tied to parallelization
  4. really hard at $O(10^6)$ processes

# Decomposition Techniques

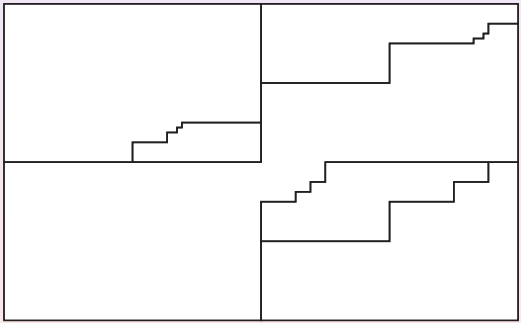
| | Com. Volume | Overhead | Load Balancing |
|---|---|---|---|
| checkerboarding | $\oplus$ | $\oplus$ | $\ominus$ |

- Com. Volume = Communication Volume
- $\oplus$ = good, $\odot$ = medium, $\ominus$ = bad



# Decomposition Techniques

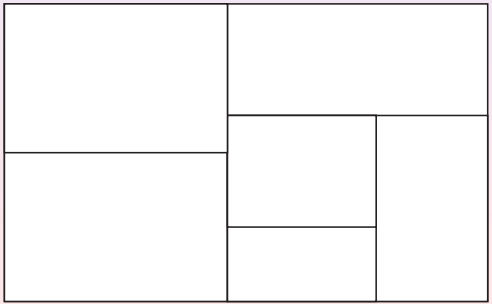
| | Com. Volume | Overhead | Load Balancing |
|---|---|---|---|
| checkerboarding | $\oplus$ | $\oplus$ | $\ominus$ |
| space filling curves | $\odot$ | $\odot$ | $\oplus$ |



# Decomposition Techniques

| | Com. Volume | Overhead | Load Balancing |
|---|---|---|---|
| checkerboarding | ⊕ | ⊕ | ⊖ |
| space filling curves | ⊙ | ⊙ | ⊕ |
| WRCB | ⊕ | ⊕ | ⊙ |

- WRCB = Weighted Recursive Coordinate Bisection

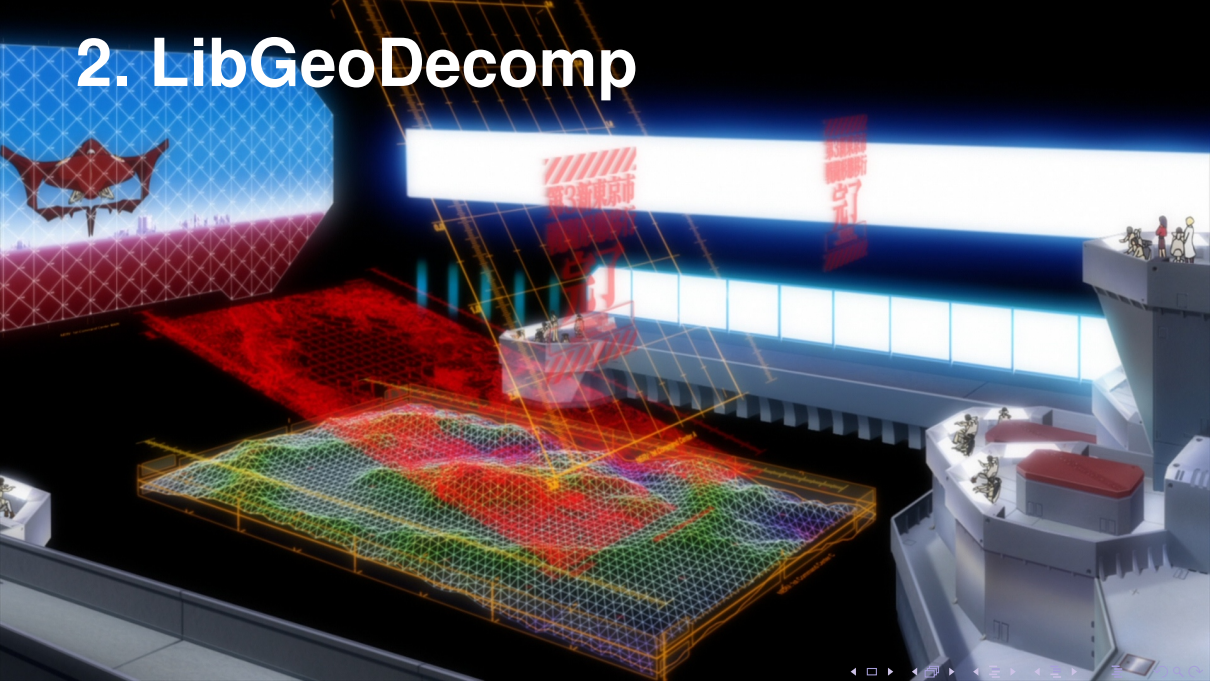


# Decomposition Techniques

| | Com. Volume | Overhead | Load Balancing |
|---|---|---|---|
| checkerboarding | ⊕ | ⊕ | ⊖ |
| space filling curves | ⊙ | ⊙ | ⊕ |
| WRCB | ⊕ | ⊕ | ⊙ |
| graph partitioners | ⊕ | ⊖ | ⊕ |

- graph partitioners = JOSTLE, ParMETIS etc.

# 2. LibGeoDecomp



# Library for Geometric Decomposition codes

- library for computer simulations
- supported models:
  - stencil codes
  - particle-in-cell codes
  - short-ranged n-body codes
  - meshfree codes

# Library for Geometric Decomposition codes

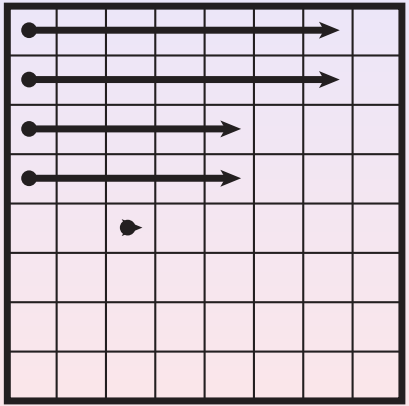
- library for computer simulations
- supported models:
  - stencil codes
  - particle-in-cell codes
  - short-ranged n-body codes
  - meshfree codes
- supported architectures:
  - Android, PC...
  - GPUs (tested on Tsubame 2.0)
  - Intel MIC (tested on Stampede)
  - Blue Gene/Q (tested on JUQUEEN)

# Library for Geometric Decomposition codes

- library for computer simulations
- supported models:
  - stencil codes
  - particle-in-cell codes
  - short-ranged n-body codes
  - meshfree codes
- supported architectures:
  - Android, PC...
  - GPUs (tested on Tsubame 2.0)
  - Intel MIC (tested on Stampede)
  - Blue Gene/Q (tested on JUQUEEN)
- flexible geometry subsystem
  - striping
  - weighted recursive coordinate bisection
  - space filling curves (Hilbert, H-Indexing, Z-Curve)

# Domain Decomposition in LibGeoDecomp

- load adaptation via weight vector
  (adapts to machine and model hotspots)
- `Partition`
  - input: weight vector, rank
  - output: domain (set of coordinates)
- `Region`:
  - set of coordinates
  - run-length compression
  - supported operations:
    - union, cut-set, subtraction
    - expansion
    - iteration
- `PartitionManager`
  - detects ghost zones
  - how to scale? (complexity: $O(n)$ vs. $O(n^2)$)



# Latency Hiding Strategies

1. overlapping communication and calculation
   $t_{total} = max(t_{calc}, t_{latency} + t_{transfer})$

# Latency Hiding Strategies

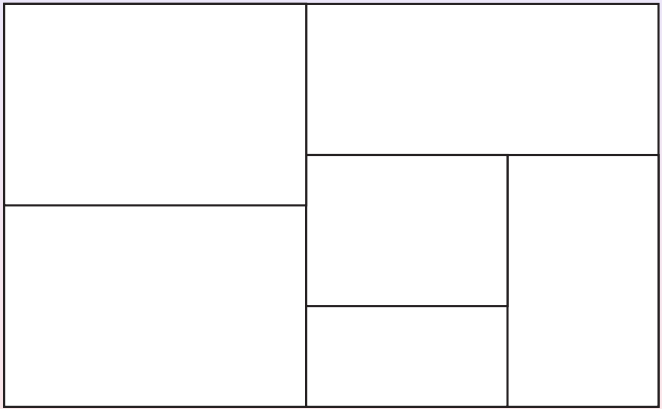
1. overlapping communication and calculation
   $t_{total} = max(t_{calc}, t_{latency} + t_{transfer})$
2. wide halos (width $k$ = communication every $k$-th time step)
   $t_{total} = t_{calc} + t_{latency}/k + t_{transfer} + t_{overhead}$

# Latency Hiding Strategies

1. overlapping communication and calculation
   $t_{total} = max(t_{calc}, t_{latency} + t_{transfer})$
2. wide halos (width $k$ = communication every $k$-th time step)
   $t_{total} = t_{calc} + t_{latency}/k + t_{transfer} + t_{overhead}$
3. best: overlapping + wide halos
   $t_{total} = max(t_{calc}, t_{latency}/k + t_{transfer}) + t_{overhead}$

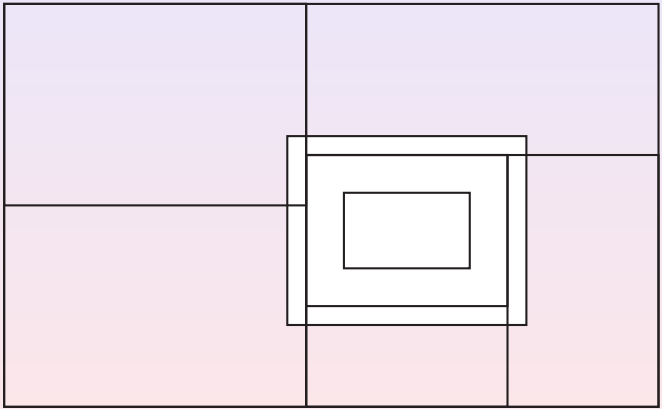
# Overlapping Com./Calc. + Wide Halos

- example:
  ghost zone width = 3



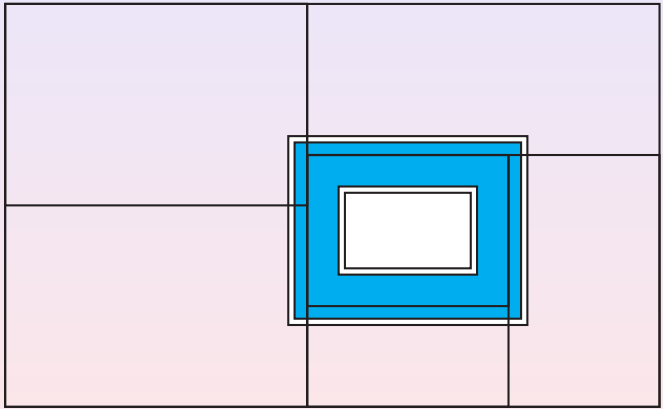
# Overlapping Com./Calc. + Wide Halos

- initial condition



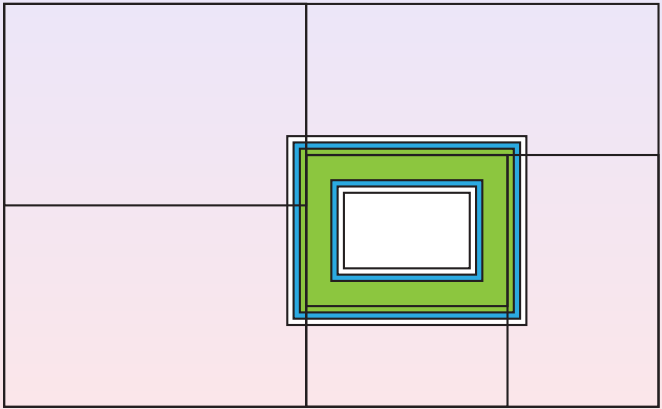
# Overlapping Com./Calc. + Wide Halos

1. update ghost zone (1/3)



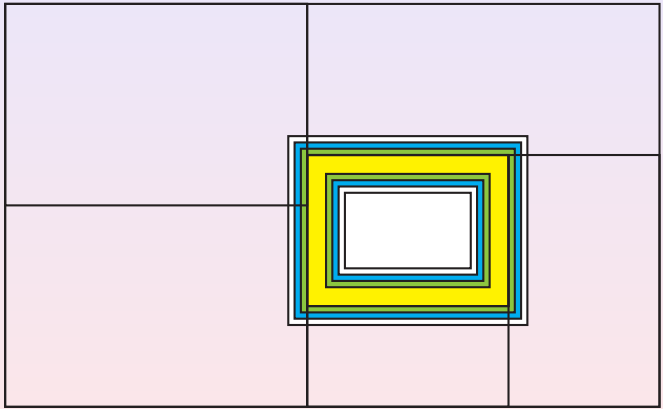
# Overlapping Com./Calc. + Wide Halos

1. update ghost zone (2/3)



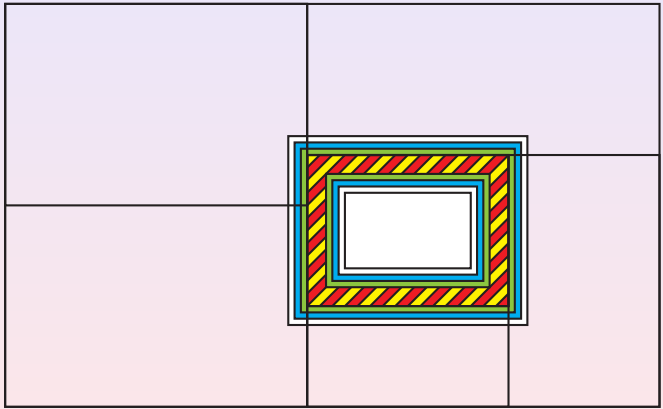
# Overlapping Com./Calc. + Wide Halos

1. update ghost zone (3/3)



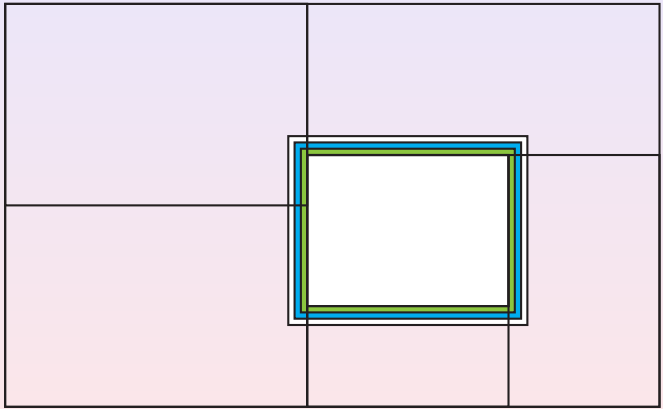
# Overlapping Com./Calc. + Wide Halos

1. update ghost zone
2. send ghost zones



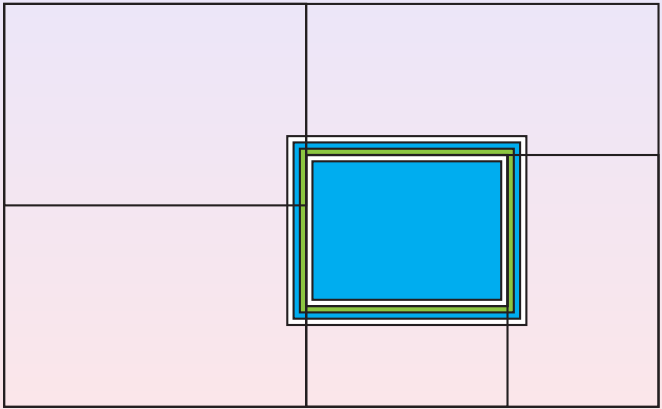
# Overlapping Com./Calc. + Wide Halos

1. update ghost zone
2. send ghost zones
3. restore inner ghost zone



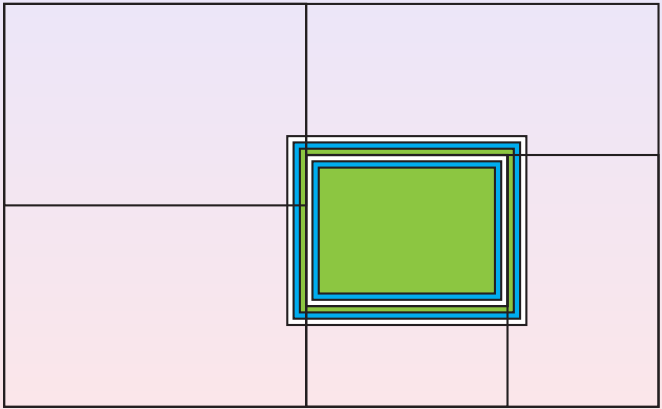
# Overlapping Com./Calc. + Wide Halos

1. update ghost zone
2. send ghost zones
3. restore inner ghost zone
4. update interior (1/3)



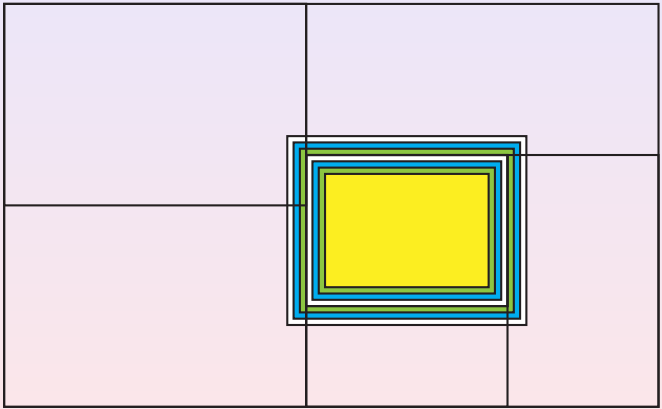
# Overlapping Com./Calc. + Wide Halos

1. update ghost zone
2. send ghost zones
3. restore inner ghost zone
4. update interior (2/3)



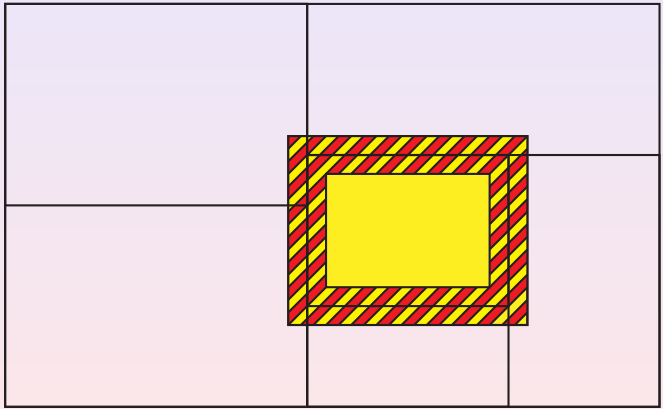
# Overlapping Com./Calc. + Wide Halos

1. update ghost zone
2. send ghost zones
3. restore inner ghost zone
4. update interior (3/3)



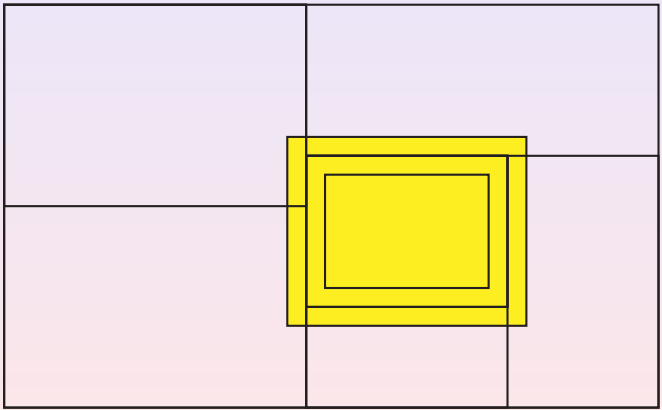
# Overlapping Com./Calc. + Wide Halos

1. update ghost zone
2. send ghost zones
3. restore inner ghost zone
4. update interior
5. restore inner/outer ghost

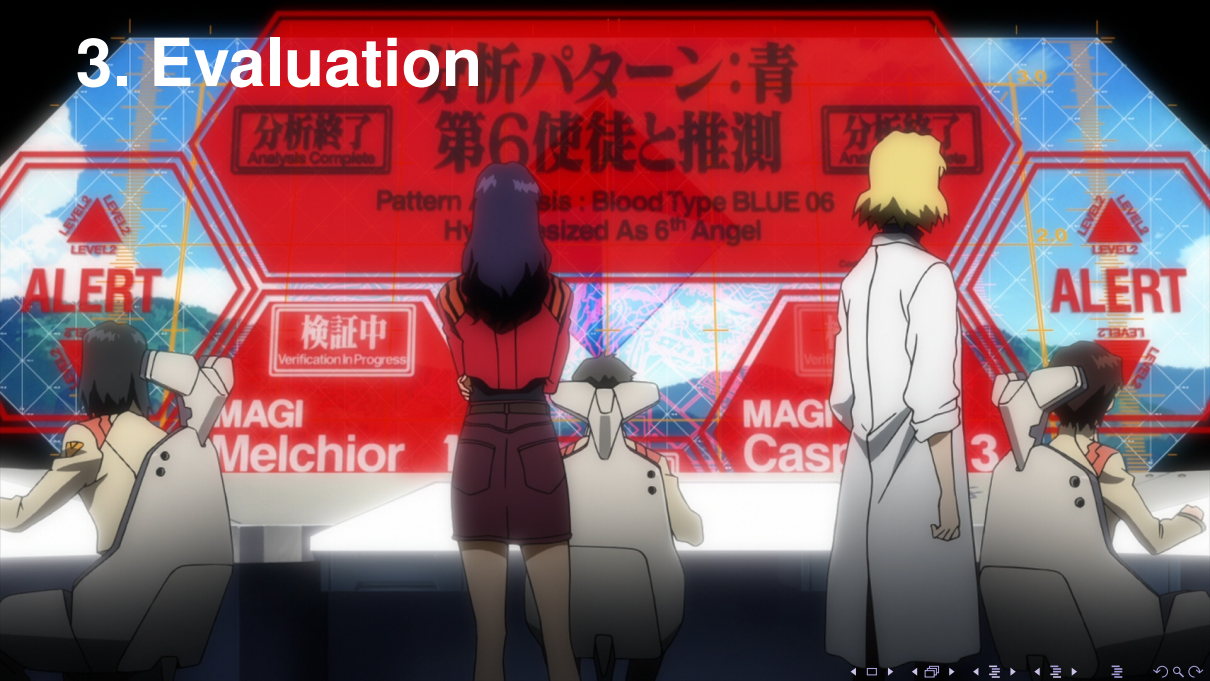


# Overlapping Com./Calc. + Wide Halos

1. update ghost zone
2. send ghost zones
3. restore inner ghost zone
4. update interior
5. restore inner/outer ghost (wait for communication)

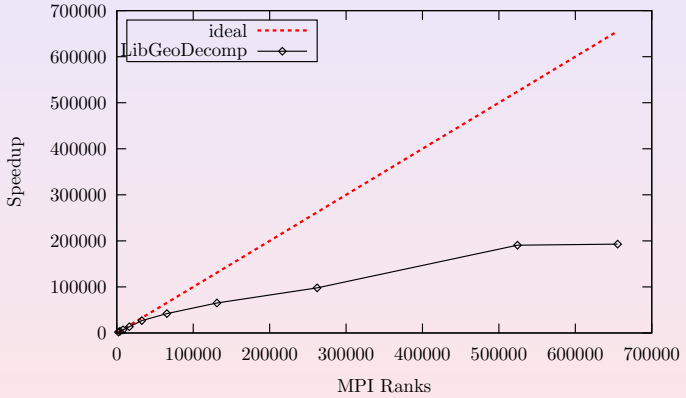


# 3. Evaluation



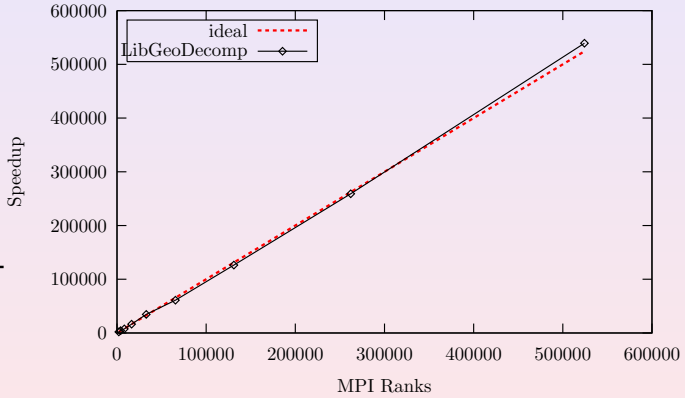
# Strong Scaling of N-Body Code on Juqueen

- scaled up to 655k MPI ranks
- fixed at 90M particles
- decomposition: recursive bisection
- great scalability:
  - load of 1 rank split among 655k ranks
  - still at 37 % of optimum (at 524k ranks)



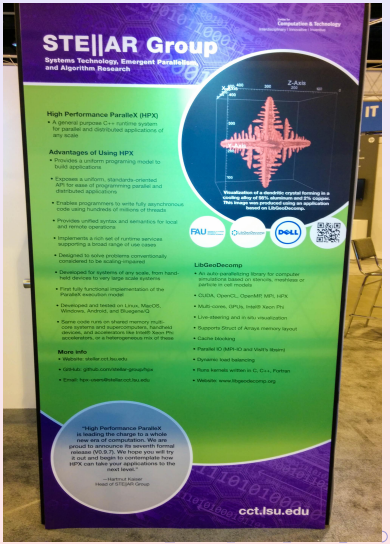
# Weak Scaling of N-Body Code on Juqueen (cont.)

- scaled up to 524k MPI ranks (1.85M run: different parameters)
- up to 234.722 Giga Particles
- decomposition: recursive bisection
- 4x oversubscription to utilize 4x SMT



# Summary

- LibGeoDecomp
  - architectures: smartphone to supercomputer
  - models: stencil codes to short-ranged n-body...
  - modular architecture
- flexible geometry subsystem
  - adapts to model, machine
- tomorrow: release 0.3.0
- outlook: extended measurements on JUQUEEN, Titan, Stampede, SuperMUC
- live demo: booth #1901 (STEllAR group, LSU)




KONWIHR

JÜLICH FORSCHUNGSZENTRUM