HPI Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Data Partitioning Strategies for Stencil Computations on NUMA Systems

Frank Feinbube, *Max Plauth*, Marius Knaust, Andreas Polze
Operating Systems and Middleware Group
Hasso Plattner Institute, University of Potsdam

Who are we?

Operating Systems and Middleware Group

- Group leader: Prof. Dr. Andreas Polze
- 8 PhD students
- „Extending the reach of Middleware"

Sanssouci Palace, Potsdam

HPI Main Campus

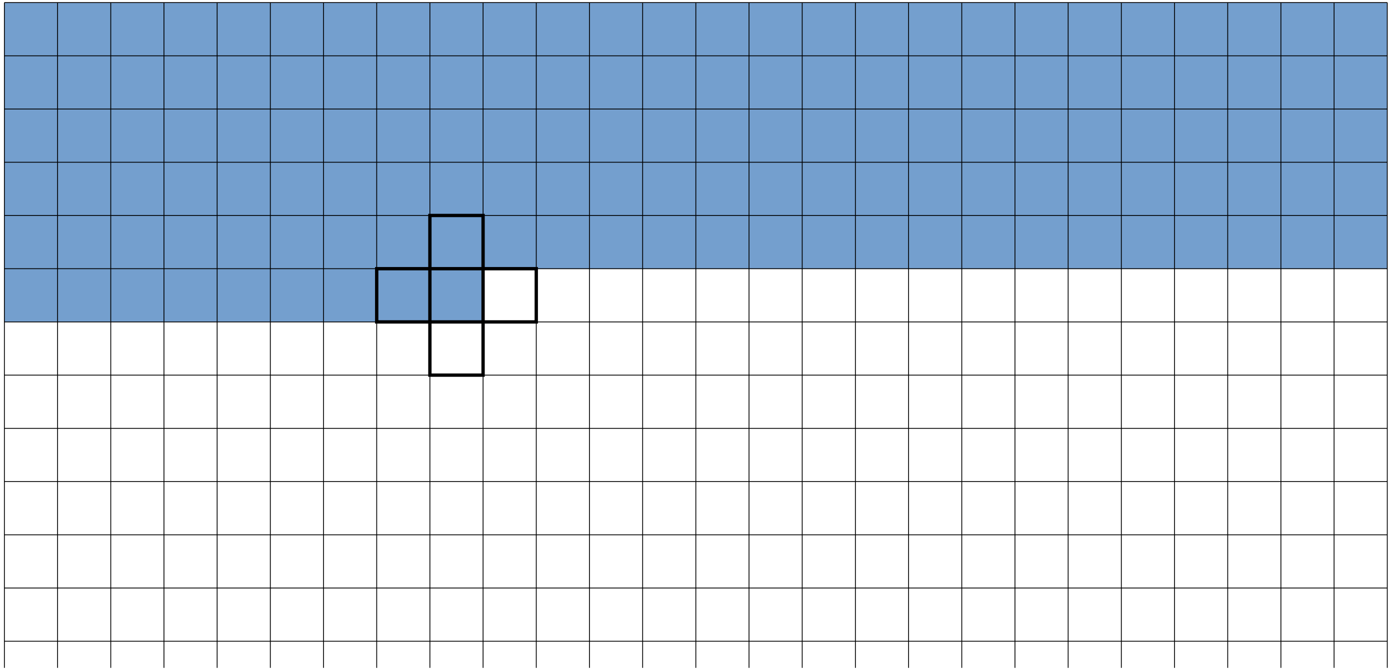
Outline

1. **Background**
2. Research Question & Contributions
3. Approaches
   - Evolutionary Partitioning Technique
   - Geometric Partitioning Technique
4. Theoretical Analysis
5. Practical Evaluation
6. Conclusion

HPI Hasso Plattner Institut

Data Partitioning Strategies for **Stencil Computations** on NUMA Systems

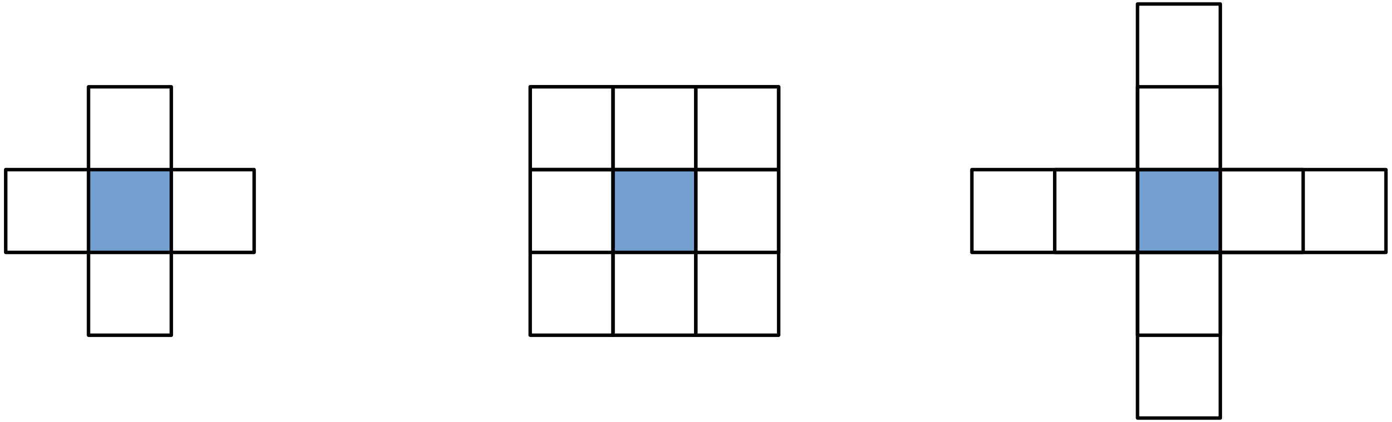
Stencils := Iterative Kernels



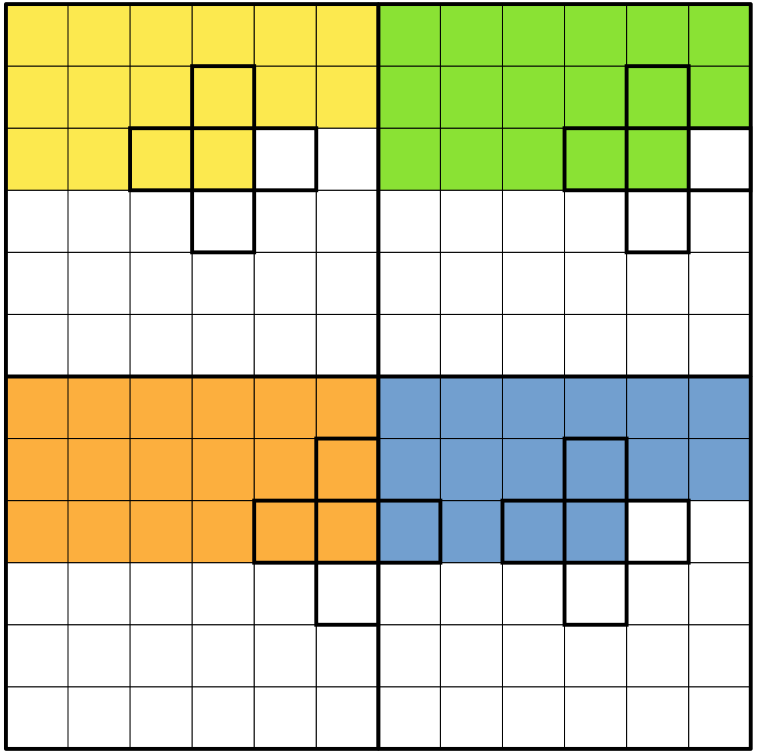
Data Partitioning Strategies for Stencil Computations on NUMA Systems

Max Plauth, 28.08.2017

Stencil Shapes

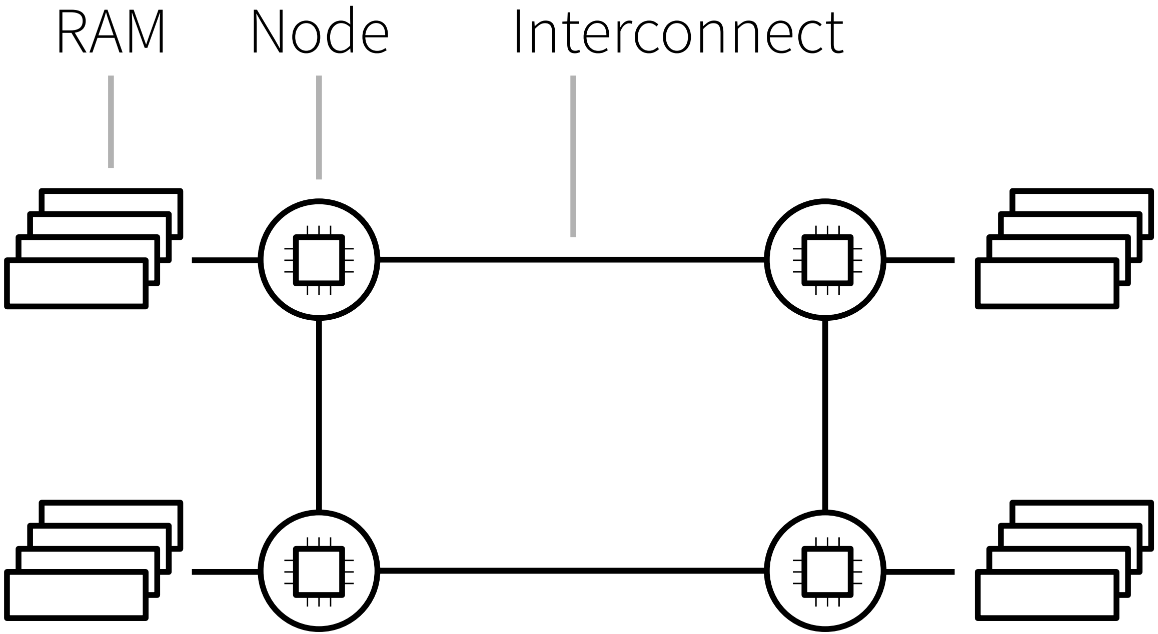


Parallel Stencil Computation

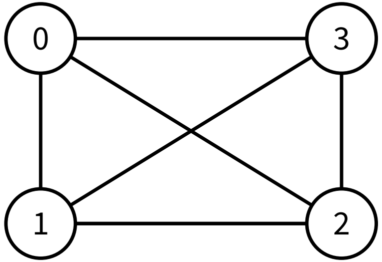


Data Partitioning Strategies for Stencil Computations on **NUMA Systems**

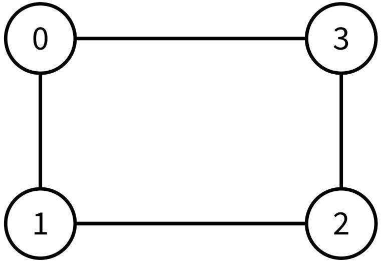
NUMA Systems



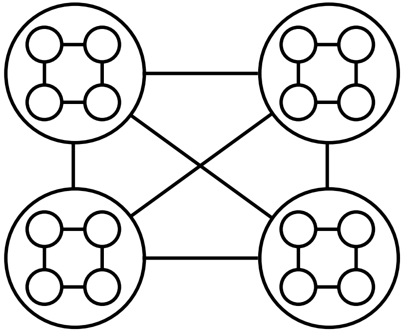
NUMA Topologies



Fully Connected



Connected

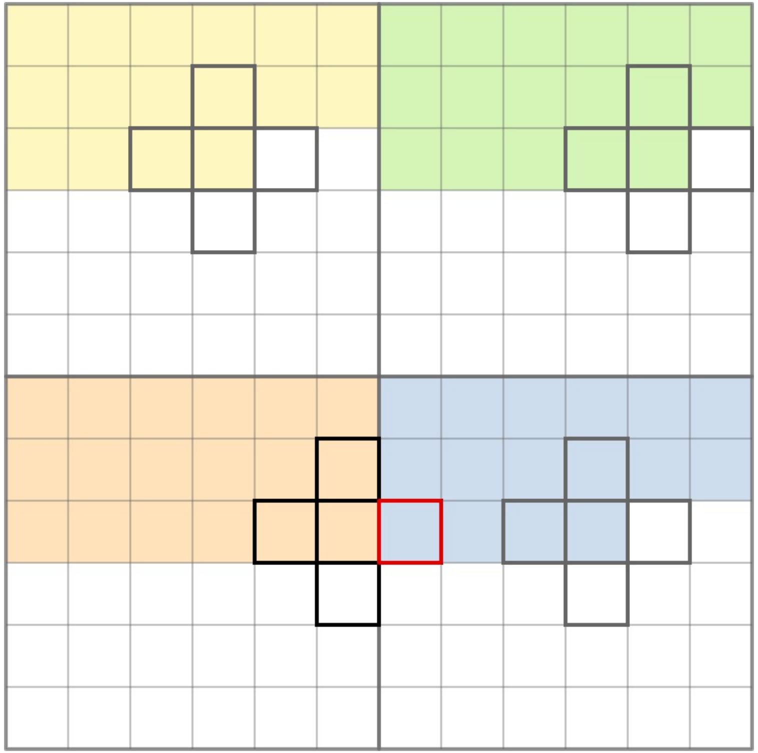


Hierarchical

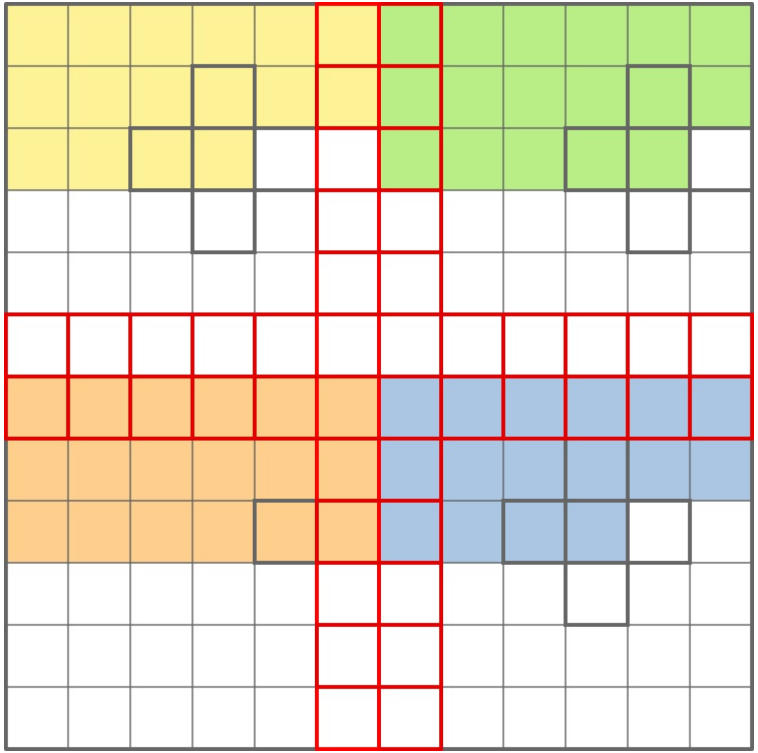
Hasso Plattner Institut

Data Partitioning Strategies for **Stencil Computations on NUMA Systems**

Stencil Computations on NUMA Systems



Stencil Computations on NUMA Systems



Outline

1. Background
2. **Research Question & Contributions**
3. Approaches
   - Evolutionary Partitioning Technique
   - Geometric Partitioning Technique
4. Theoretical Analysis
5. Practical Evaluation
6. Conclusion

Research Question & Contributions

- Research Question:
  - *"This work aims at finding partitioning strategies that reduce the occurrence of remote memory access on modern NUMA systems."*

- Contribution
  - Based on evolutionary algorithms, a partitioning approach is presented.
  - A geometric partitioning strategy is developed to overcome the limitations of the evolutionary approach.
  - The retrieved strategies are elucidated from a theoretical perspective.
  - A practical evaluation on a real hardware shows that the number of remote memory accesses can indeed be decreased with the presented approaches.

Outline

1. Background
2. Research Question & Contributions
3. **Approaches**
   - Evolutionary Partitioning Technique
   - Geometric Partitioning Technique
4. Theoretical Analysis
5. Practical Evaluation
6. Conclusion

Evolutionary Approach

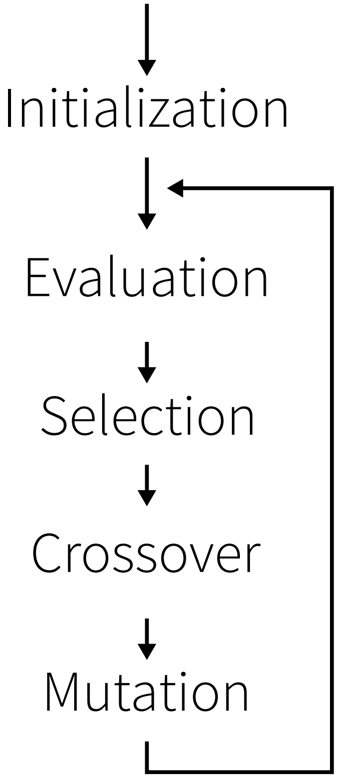
Input Data for Evolutionary Approach

- Grid Properties
  - Grid resolution (also with different side ratios)
  - Cell types
- Access Pattern
  - Any stencil (as code)
  - Other kernels (with multiple inputs)
- System Configuration
  - Remote access cost matrix
  - Cache sizes

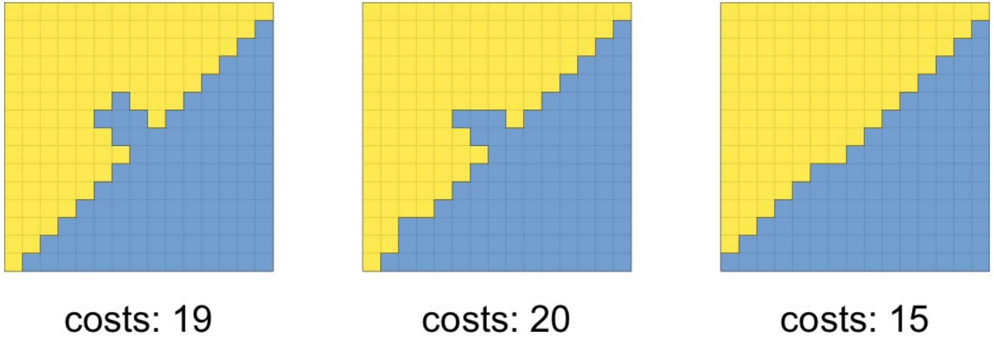
Example Usage

```
using Data = Matrix<unsigned, sideLength, sideLength>;

auto fivePoint = [](size_t x, size_t y, const Data &input)
     {
          if (y >= 1) input(x, y - 1);
          if (x >= 1) input(x - 1, y);
          if (y < Data::sizeX() - 1) input(x, y + 1);
          if (x < Data::sizeY() - 1) input(x + 1, y);
     };

Costs costHPProLiantDL980G7
     {
          {10, 12, 17, 17, 19, 19, 19, 19},
          {12, 10, 17, 17, 19, 19, 19, 19},
          {17, 17, 10, 12, 19, 19, 19, 19},
          {17, 17, 12, 10, 19, 19, 19, 19},
          {19, 19, 19, 19, 10, 12, 17, 17},
          {19, 19, 19, 19, 12, 10, 17, 17},
          {19, 19, 19, 19, 17, 17, 10, 12},
          {19, 19, 19, 19, 17, 17, 12, 10}
     };

Evolution<Data, 1000> evolution(fivePoint, costHPProLiantDL980G7);
```

General Procedure & Optimization Strategies



- Elitist Selection
  - Add parent individual to the child generation
- Escaping Local Minima with Multiple Changes
  - Keep the changes local to each other
- Resets



Results (Evolutionary Technique)

1	1	1	1	1	1	1	1	0	0
1	1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0

(2) costs: 20

1	1	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	2
0	0	0	0	0	1	1	1	2	2
0	0	0	0	0	0	2	2	2	2
0	0	0	0	0	2	2	2	2	2
0	0	0	0	2	2	2	2	2	2
0	0	0	2	2	2	2	2	2	2
0	0	2	2	2	2	2	2	2	2

(3) costs: 30

2	2	2	2	1	1	1	1	1	1
2	2	2	2	2	1	1	1	1	1
2	2	2	2	2	2	1	1	1	1
2	2	2	2	2	2	1	1	1	1
0	0	2	2	2	3	3	1	1	1
0	0	0	2	3	3	3	3	1	1
0	0	0	0	3	3	3	3	3	1
0	0	0	0	0	3	3	3	3	3
0	0	0	0	0	3	3	3	3	3
0	0	0	0	0	0	3	3	3	3

(4) costs: 37

1	1	1	1	1	1	3	3	3	3
1	1	1	1	1	3	3	3	3	3
1	1	1	1	1	3	3	3	3	3
1	1	1	0	0	0	3	3	3	3
2	1	0	0	0	0	0	3	3	4
2	2	0	0	0	0	0	0	4	4
2	2	2	0	0	0	0	4	4	4
2	2	2	2	0	0	4	4	4	4
2	2	2	2	2	4	4	4	4	4
2	2	2	2	2	4	4	4	4	4

(5) costs: 45

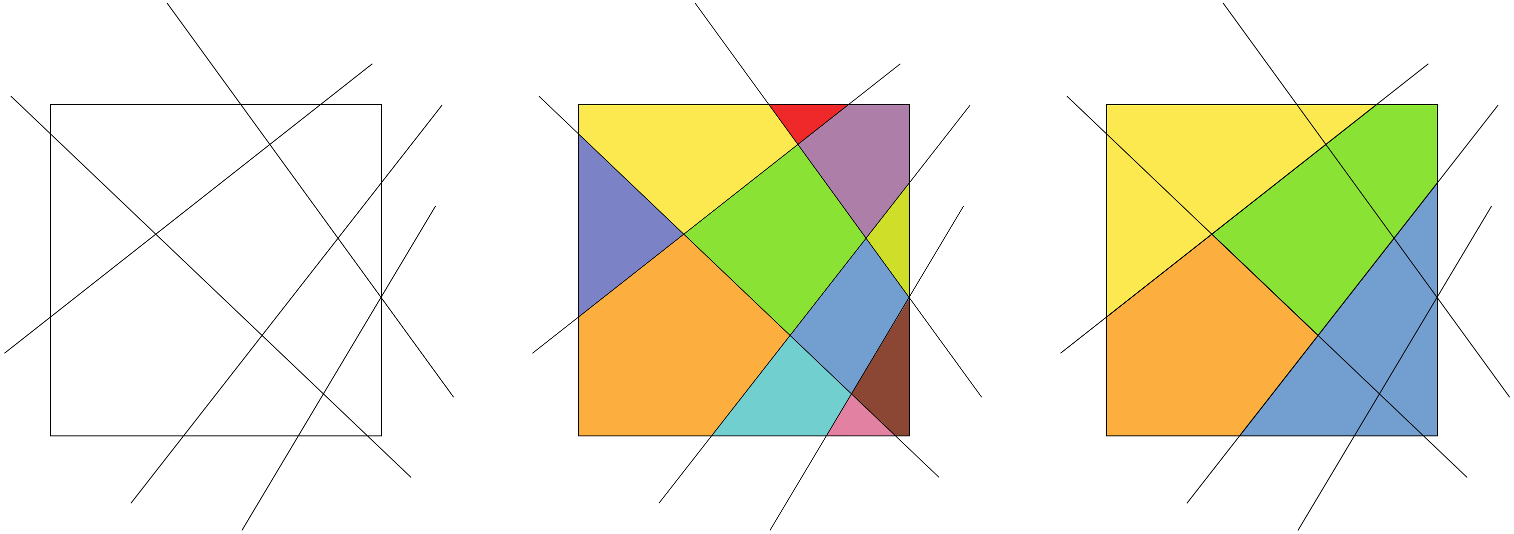
Drawbacks

- Limited to small NUMA node counts
  - More NUMA nodes require a higher resolution

- Exploding search space
  - The search space grows quadratic with the side length.
  - Severely limited feasibility already at node counts with $n > 4$

Geometric Approach

Geometric Algorithm

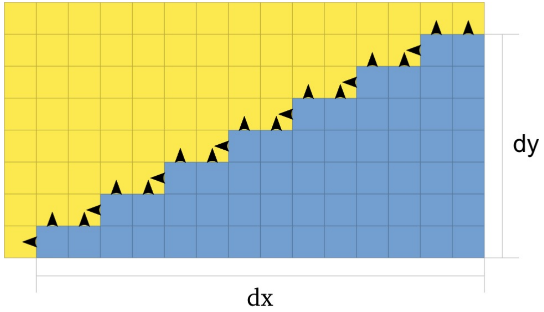


Score Function

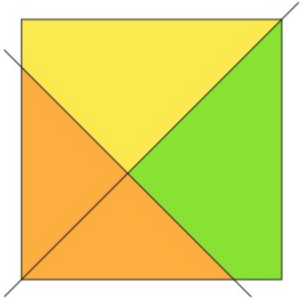
- Optimize for cost and area difference
  - There is no guarantee that all partition shapes have the same area

$$\text{score} = \text{cost} * \frac{\text{area}_{\text{max}}}{\text{area}_{\text{min}}}$$

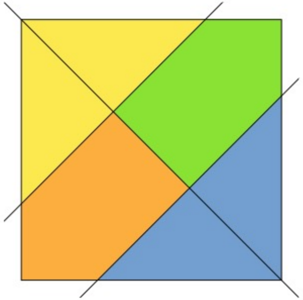
- Calculate the cached communication cost
  - The edge cost equals the maximum of the projections to the axis



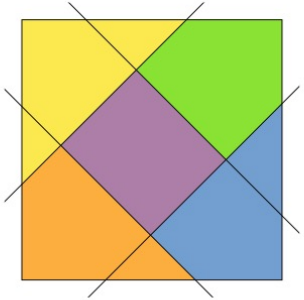
Results (Geometric Technique)



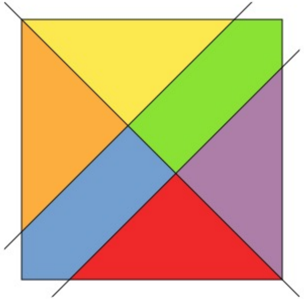
(3) costs: 2.826



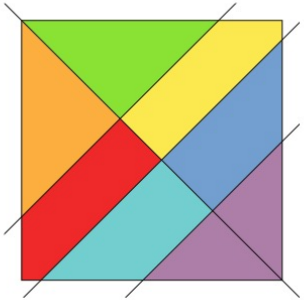
(4) costs: 3.414



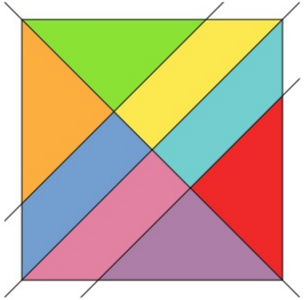
(5) costs: 4.000



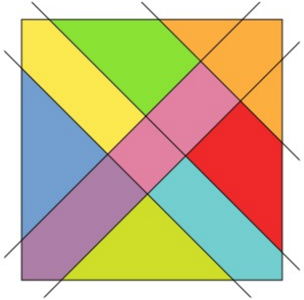
(6) costs: 5.266



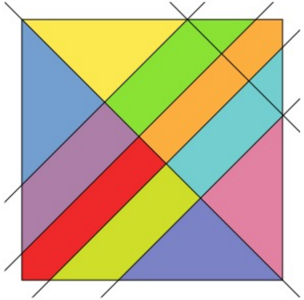
(7) costs: 5.898



(8) costs: 6.828



(9) costs: 6.804

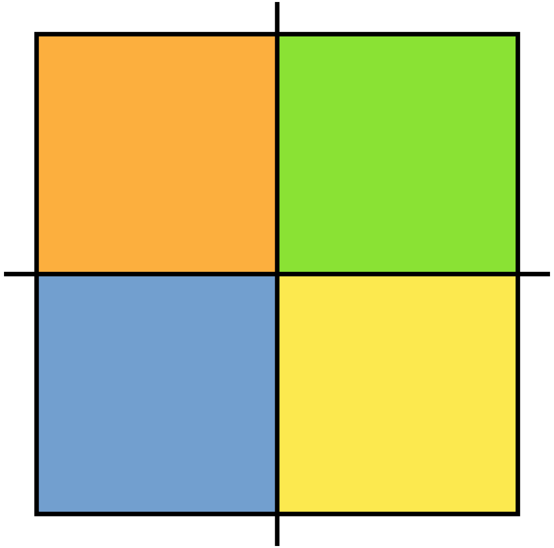


(10) costs: 7.476

Outline

1. Background
2. Research Question & Contributions
3. Approaches
   - Evolutionary Partitioning Technique
   - Geometric Partitioning Technique
4. **Theoretical Analysis**
5. Practical Evaluation
6. Conclusion

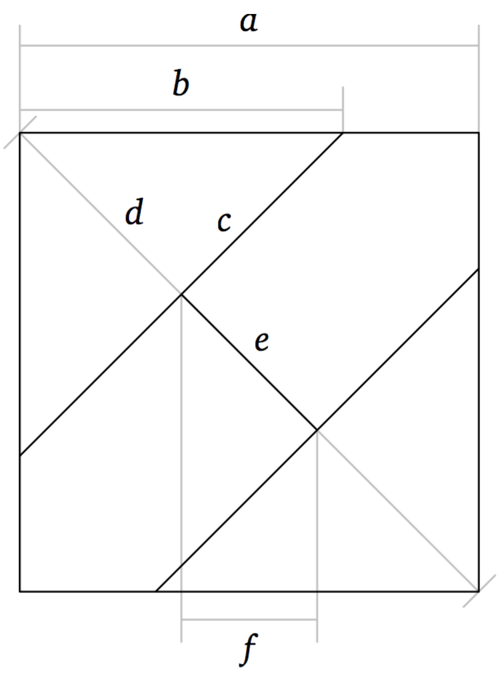
Reference: Rectangular Partitioning Strategy



$$
\begin{aligned}
U_{\text{rectangle}} &= 2(a+c) + 2(a-c) \\
&= 2a + 2c + 2a - 2c \\
&= 4a
\end{aligned}
$$

$$\text{cost} = 4a$$

Reference: Rectangular Partitioning Strategy



$$
\begin{aligned}
b &= \sqrt{\frac{a^2}{2}} &&= a\sqrt{\frac{1}{2}} \\
c &= \sqrt{b^2 + b^2} &&= a \\
d &= \sqrt{a^2 + a^2} &&= a\sqrt{2} \\
e &= d - 2\frac{c}{2} &&= a\left(\sqrt{2} - 1\right) \\
f &= \sqrt{\frac{e^2}{2}} &&= e\sqrt{\frac{1}{2}}
\end{aligned}
$$

$$\text{cost} = 2 * b + 2 * (b + f) = \left(\sqrt{2} + 2\right) a = 3.414a$$

Outline

Hasso Plattner Institut

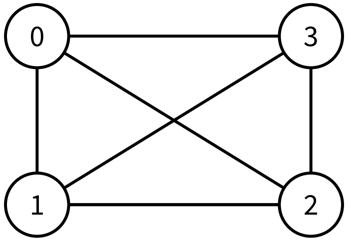
1. Background
2. Research Question & Contributions
3. Approaches
   - Evolutionary Partitioning Technique
   - Geometric Partitioning Technique
4. Theoretical Analysis
5. **Practical Evaluation**
6. Conclusion

Hypothesis & Test System

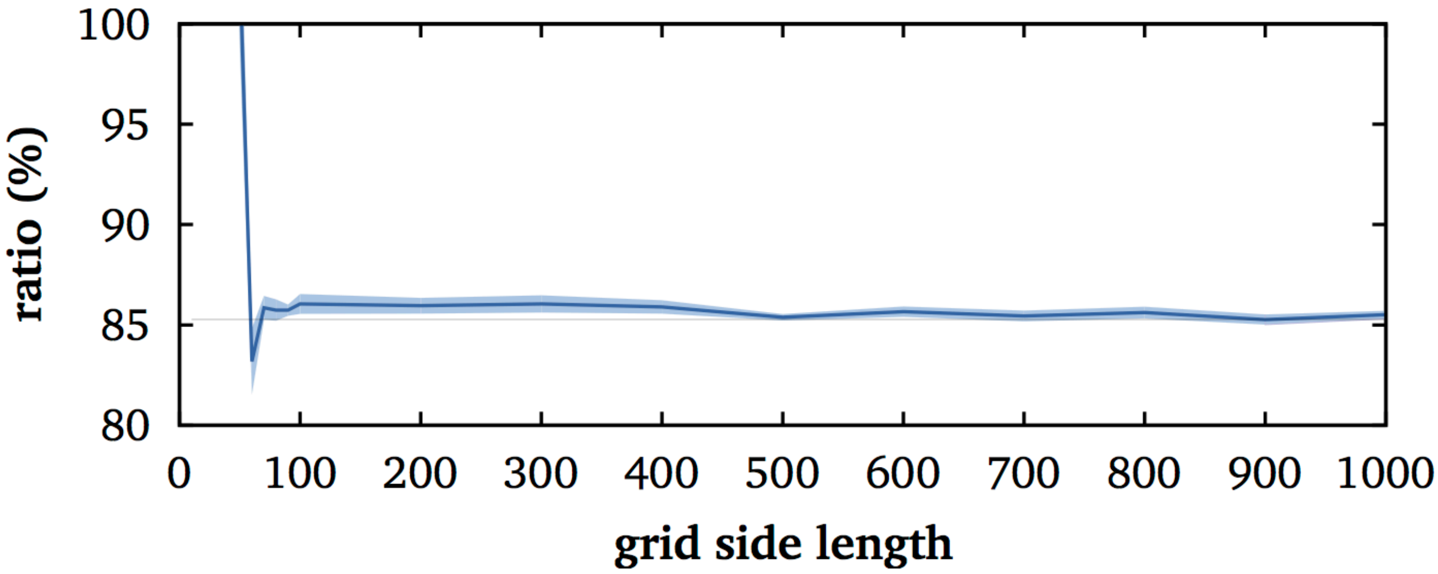
- With the geometric partitioning scheme in place, a four node system should achieve ~85% of the performance of a square partitioning layout.

$$\text{ratio} = \frac{3.414}{4} = 0.8535$$

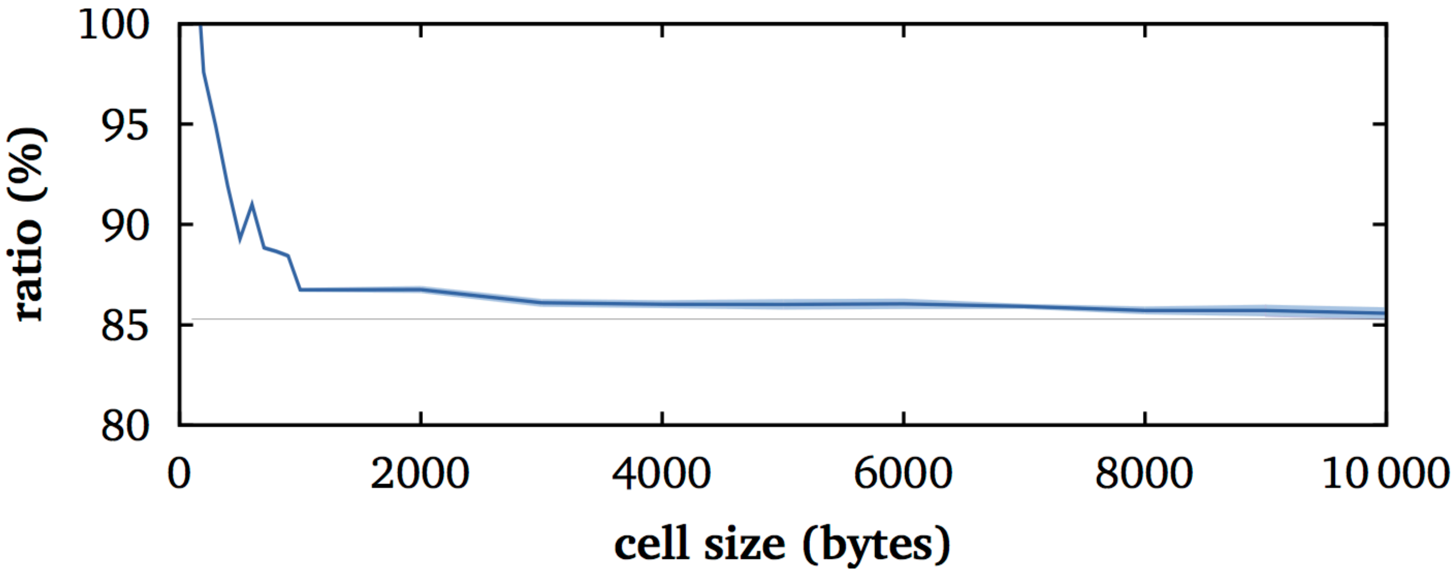
- Test System Specification: HP ProLiant DL580 G9
  - 4 x Intel Xeon E7-8890 v3 (18 cores @ 2.5 GHz)
  - 45 MB Last Level Cache
  - Each processor has its own 32 GB of memory and forms a NUMA node.



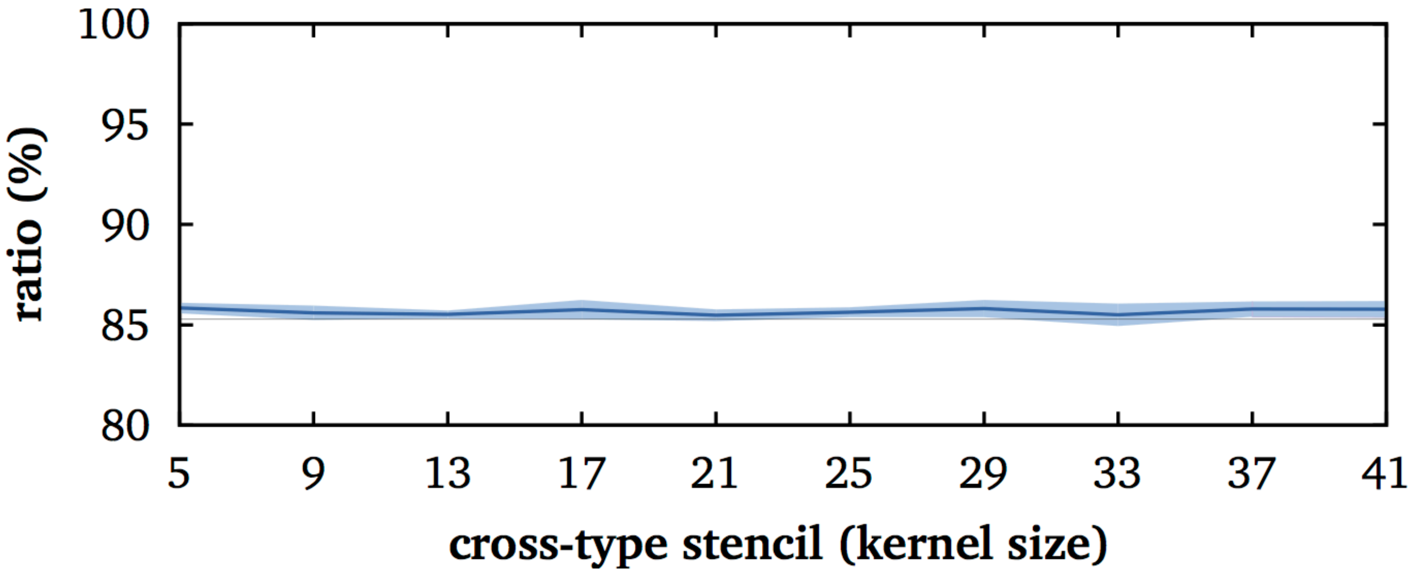
Results: Variable Grid Side Length / Fixed Cell Size



Results: Variable Cell Size / Fixed Grid Side Length



Results: Variable Cross-type Stencil Size



Outline

1. Background
2. Research Question & Contributions
3. Approaches
   - Evolutionary Partitioning Technique
   - Geometric Partitioning Technique
4. Theoretical Analysis
5. Practical Evaluation
6. **Conclusion**

Conclusion

- Partitioning strategies highly depend on the exact configuration
  - Partitioning schemes need to be tailored to the exact number of nodes.
  - Otherwise, applying the partitioning patterns could be counterproductive.

- Based on our findings, the approach seems to be suited for
  - High remote access penalties
  - Fully connected graph topologies
  - Environments without cache coherency

Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Thank You for Your Attention!

Frank Feinbube, *Max Plauth*, Marius Knaust, Andreas Polze

Operating Systems and Middleware Group

Hasso Plattner Institute, University of Potsdam