Efficient Distributed Load Balancing for Parallel Algorithms

Biagio Cosenza
Dipartimento di Informatica
Università di Salerno, Italy

Supervisor
Prof. Vittorio Scarano

Outline

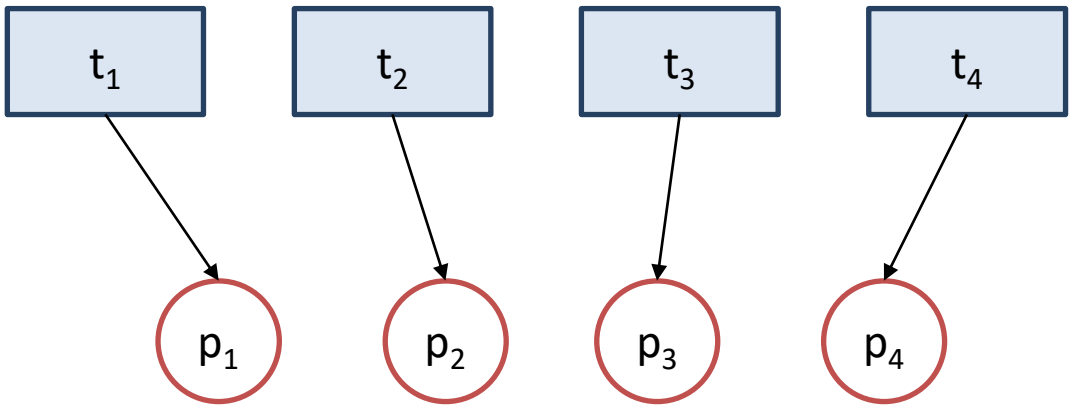
- Introduction
- The Load Balancing Problem
  - Centralized vs Distributed approach
- Scenarios
  - Mesh-based Computations
  - Parallel Ray Tracing
  - Agent-based Simulations
- Conclusion

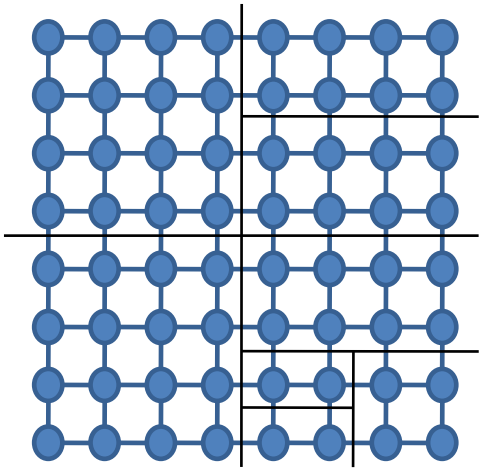
Introduction to Parallel Computing

- The need of parallel computing
- Three walls
  - Power wall
  - Memory wall
  - Instruction Level Parallelism wall
- New hardware trends
  - From multi-core to many-core
  - Low single-core frequency
- Reconsidering metrics
  - Scalability

Decomposition/Mapping

- Decomposition:
  - subdivide the computation in tasks
- Mapping
  - assing tasks to processors



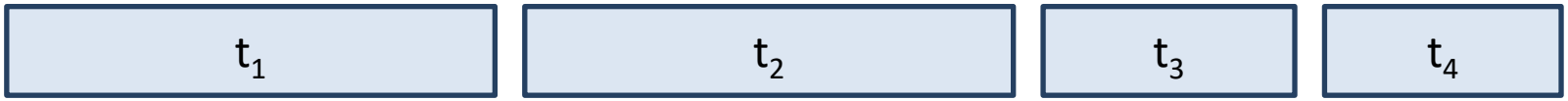


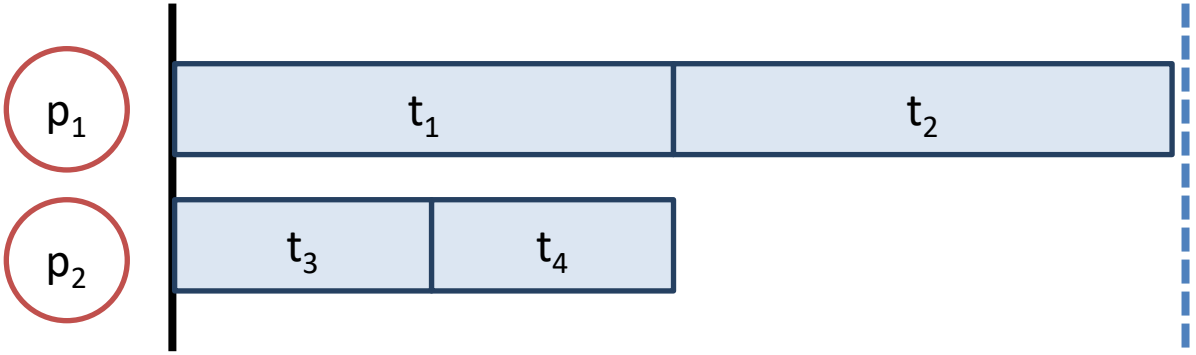
The Load Balancing Problem

- The execution time of a parallel algorithm
  - *on a given processor is determined by* the time required to perform its portion of the computation + communication/sync overhead
  - *as a whole is determined* by the longest execution time of **any** of the processors
- Balance the computation and communication between processors in such a way that the maximum per-processor execution time is minimal

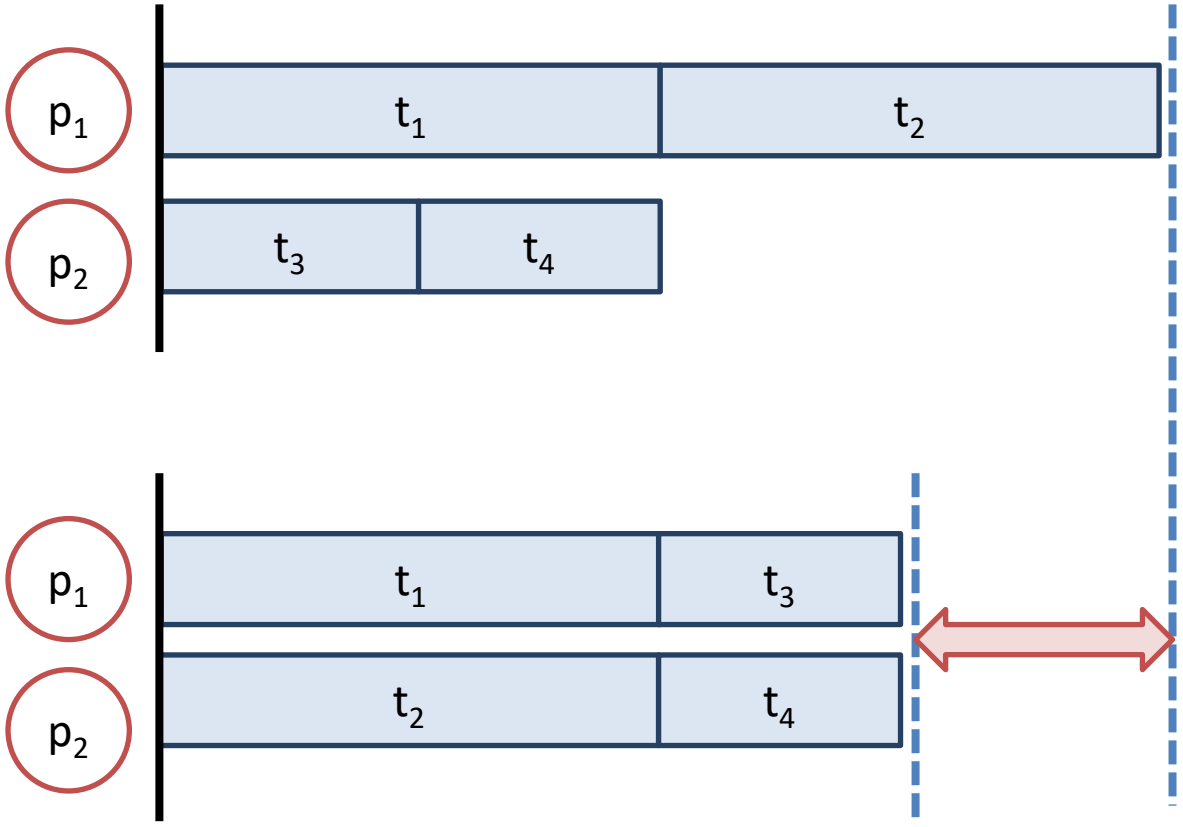
Load Balancing
An Example

- Embarassingly parallel (no dependency)
- Different per-task computation time
- 2 processors, and 4 tasks



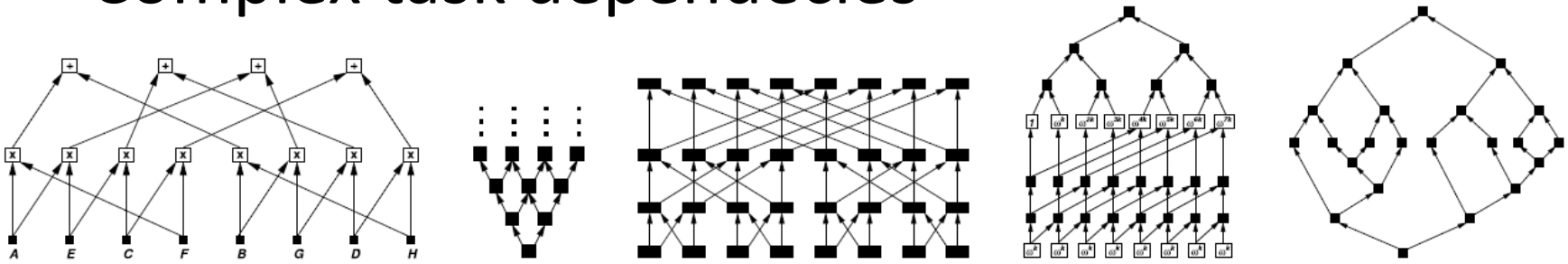


Load Balancing
An Example



Load Balancing
Non-trivial Scenario

- Complex task dependecies

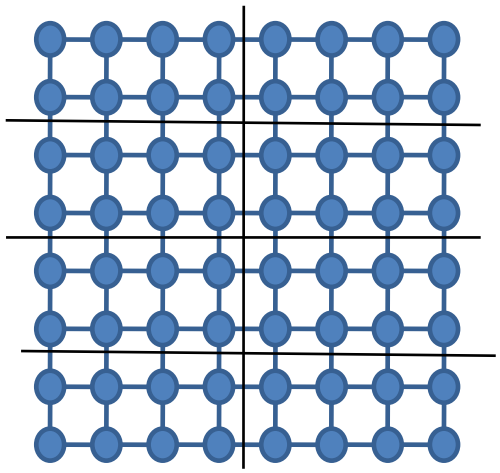


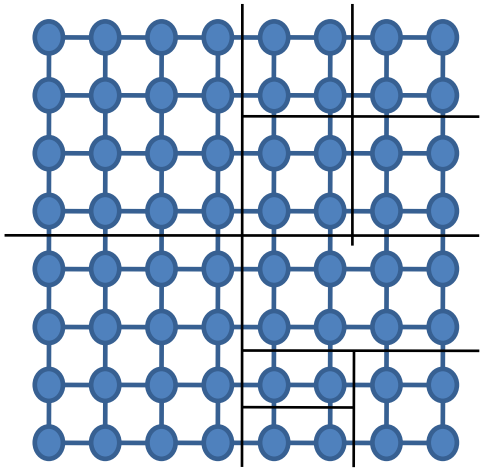
- Higher per-task compute time variance

- Computational time hard to predict

Load Balancing
Static vs Dynamic

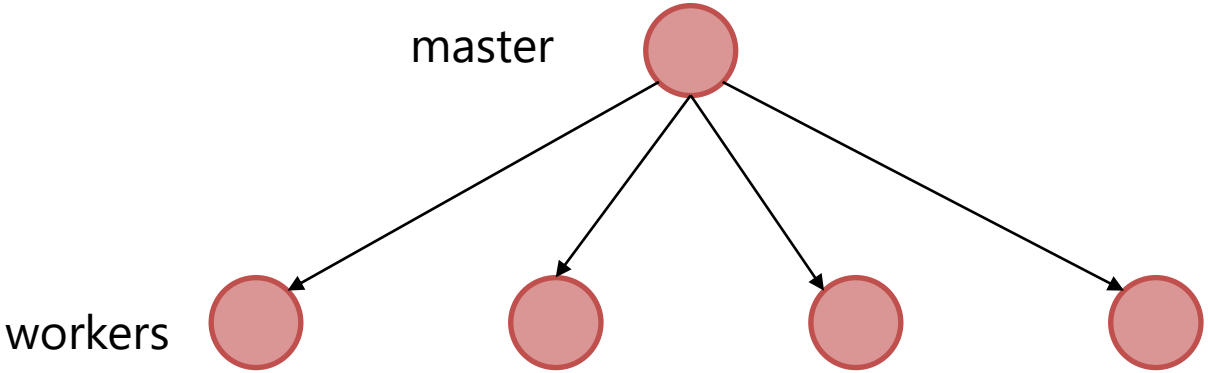
- Task distribution
  - …before execution (static)
  - …during execution (dynamic)



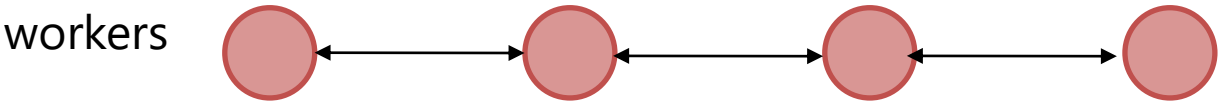


Load Balancing
Centralized vs Distributed

Centralized



Distributed



Load Balancing
Centralized vs Distributed

- Master node is a bottleneck
- Poor scalability
- Global vision of the computation
  - Smarter load balancing algorithms

- No master syncronization
- Good scalability
- Local vision of the computation
  - Load balancing algorithm can use only local informations

Scenarios

1. Mesh-based computation by using a PBT
   – centralized
   – complex use of prediction
2. Load balancing based on cost evaluation
   – use of both centralized and distributed techniques
   – complex use of prediction
3. Agent-based simulations
   – totally distributed
   – simple and fast use of prediction

1. Load Balancing on Mesh-like Computations

- Load Balancing in Mesh-like Computations using Prediction Binary Tree

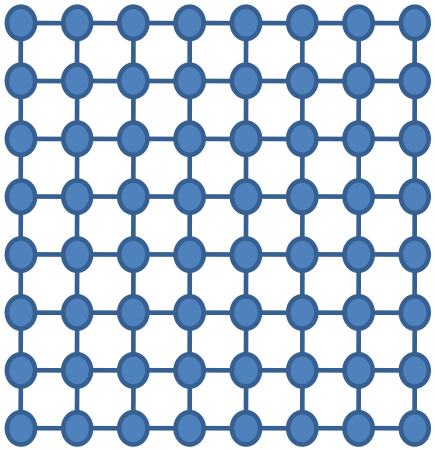
Experiences with Mesh-like computations using Prediction Binary Trees. *Gennaro Cordasco, Biagio Cosenza, Rosario De Chiara, Ugo Erra, and Vittorio Scarano.* Scalable Computing: Practice and Experience, Scientific international journa l for parallel and distributed computing (SCPE), 10(2):173–187, June 2009.

On Estimating the Effectiveness of Temporal and Spatial Coherence in Parallel Ray Tracing. *Biagio Cosenza, Gennaro Cordasco, Rosario De Chiara, Ugo Erra, and Vittorio Scarano.* In *6th Eurographics Italian Chapter Conference (EGITA 2008), July 2-4, Salerno, Italy, 2008.*

- Project HPC-Europa++
  - Tests ran at HLRS, Universitaet Stuttgart

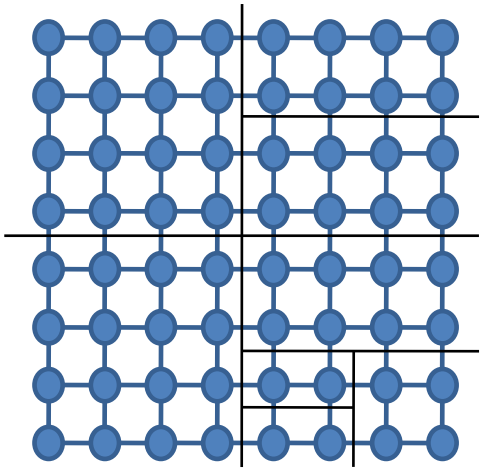
Mesh-like Computation

- A graph G=(N,E) is used to model a computation (computation-graph):
  - Each node $v \in N$ represents a task in the computation
  - Edge among nodes represents tasks dependencies
- We consider Mesh-like computation where the graph G is a 2-dimensional mesh



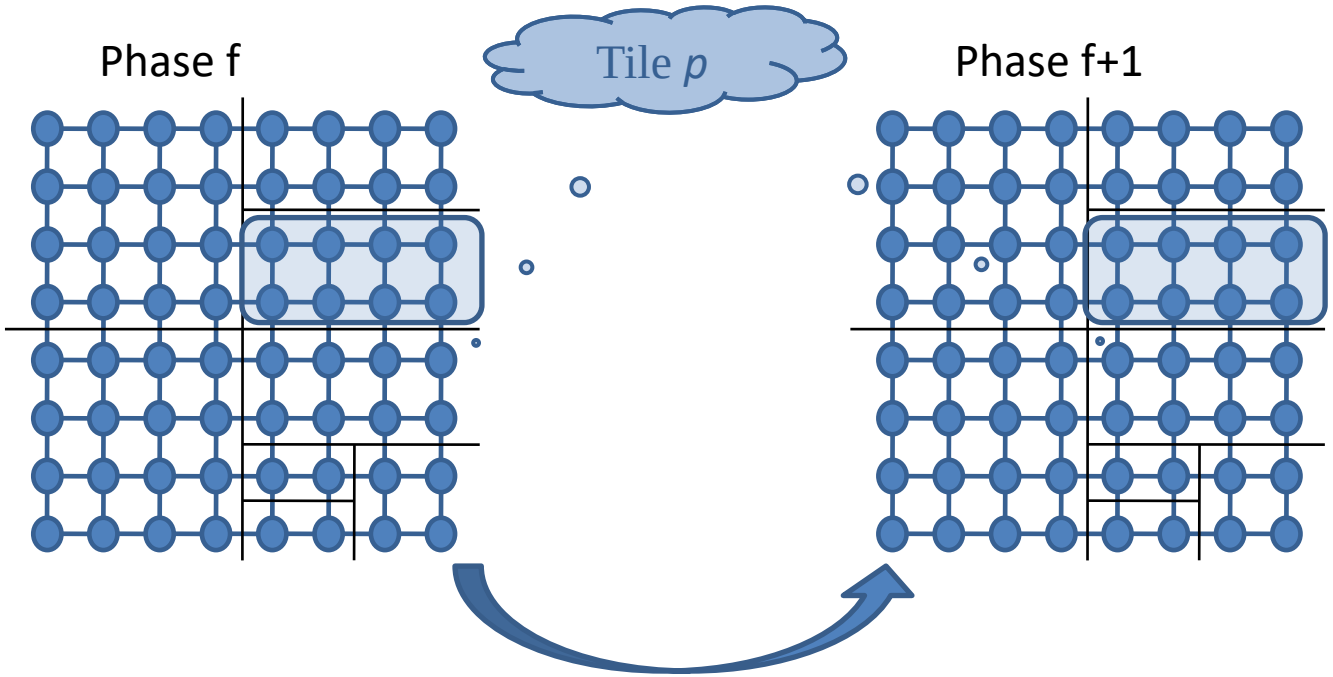
Tiled Mapping Strategies

- We are interested in tiled mapping strategies where the whole mesh is partitioned into **tiles** (contiguous 2-dimensional blocks of items).
- Tiled mapping allows to exploit the local dependencies among tasks:
  - Locality of interactions
  - Spatial coherence



Step-wise computations

- Data is computed in successive phases
  - Temporal coherence: The amount of time required by the tile *p* in phase *f* is comparable to the amount of time required by *p* in phase *f+1*

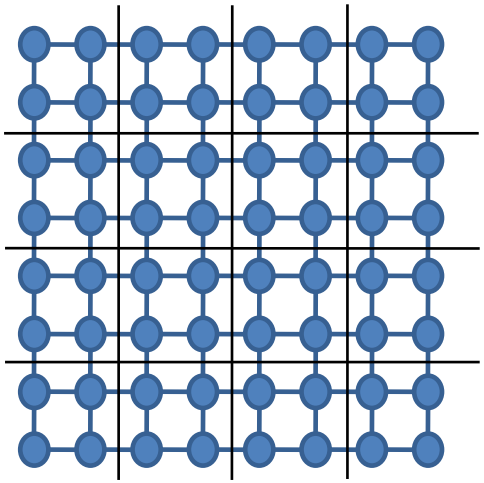


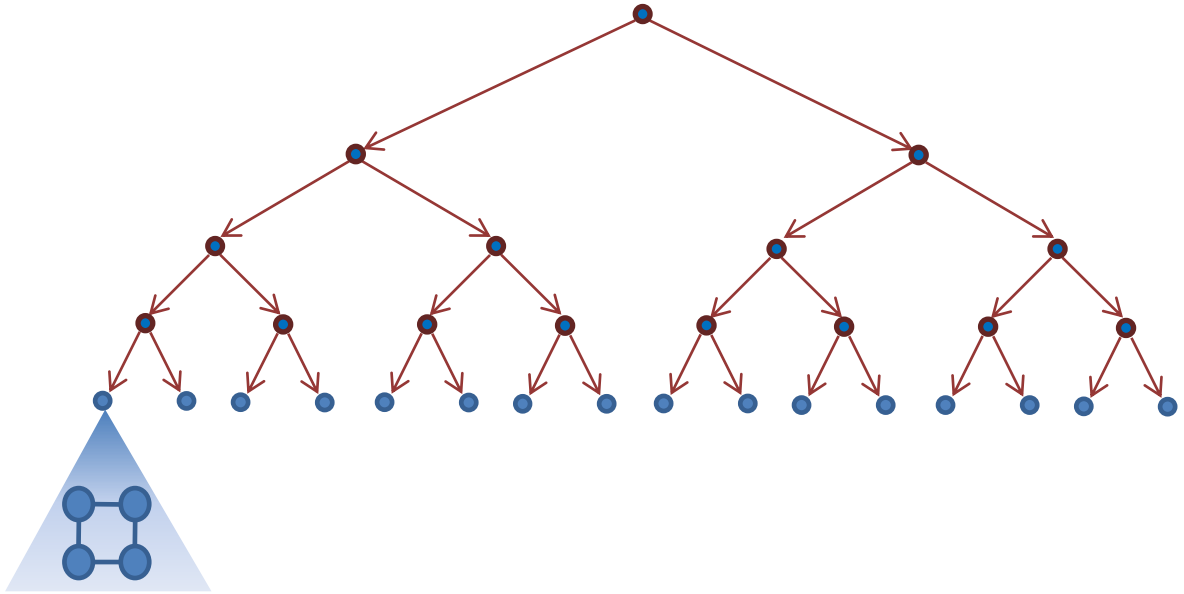
Our Proposal

- A **semi-static coarse-grained** decomposition/mapping strategy for parallel **mesh-like** computation which exploits **temporal coherence**
  - semi-static: decisions are made before each computing phase
  - coarse-grained: to reduce communication and exploit the local dependencies among tasks
  - mesh-like: the strategy is based on tiled mapping
  - temporal coherence is exploited in order to balance task sizes

The Prediction Binary Tree (PBT)

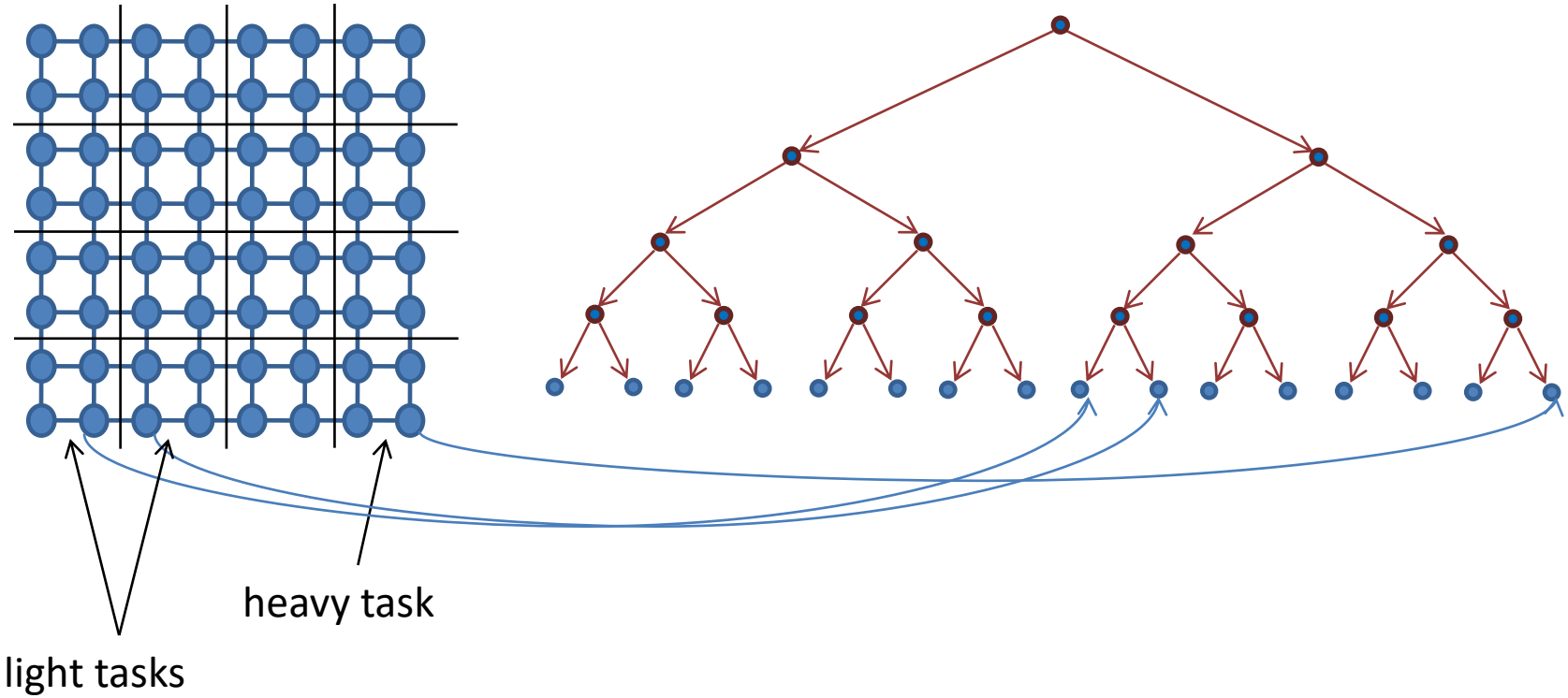
- The PBT is in charge of directing the tiling-based mapping strategy:
  - Each computing phase is split into m tiles
  - Each leaf of the PBT represents a tile





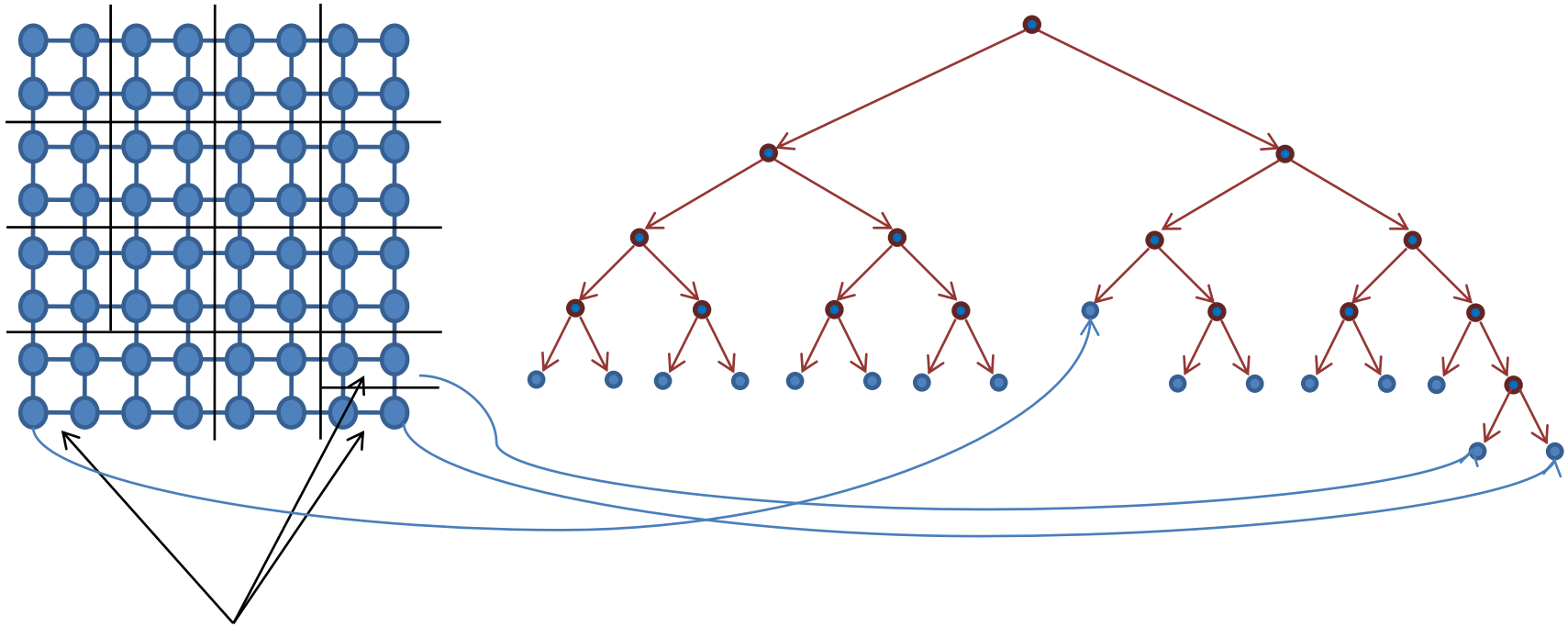
The Prediction Binary Tree (PBT)

- Before each new phase tile sizes are adjusted according to the previous phase



The Prediction Binary Tree (PBT)

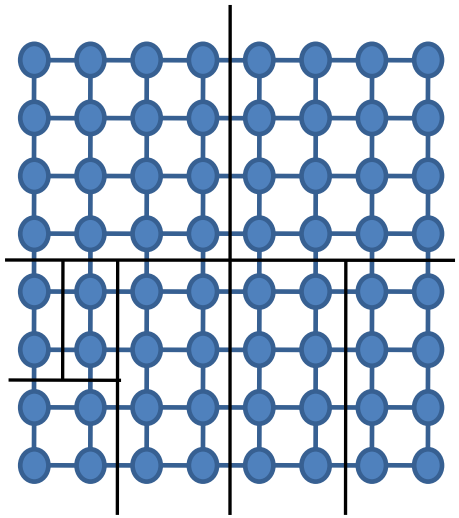
- Before each new phase tile sizes are adjusted according to the previous phase

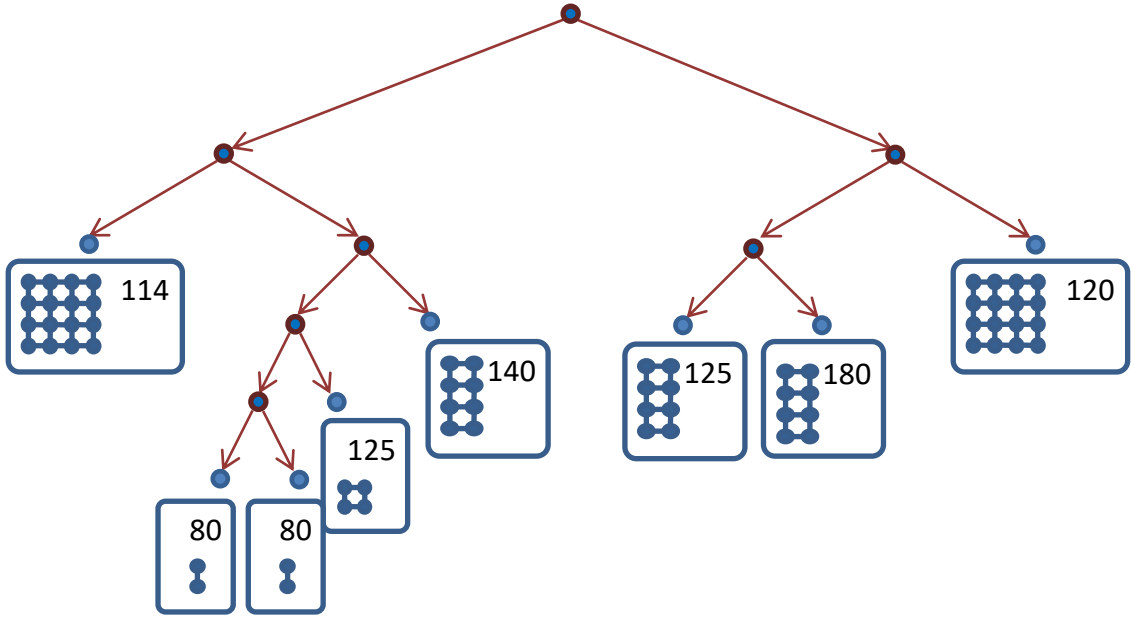


Improved balancing

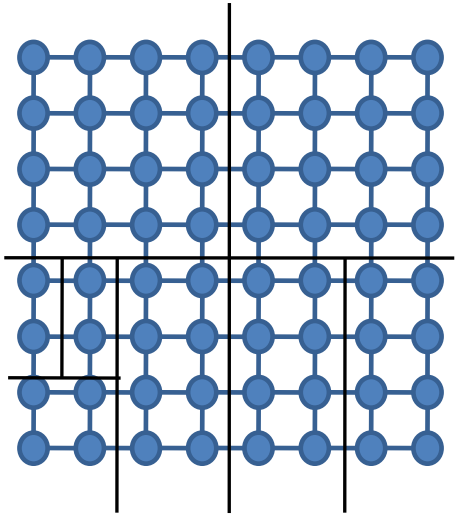
The Prediction Binary Tree (PBT)

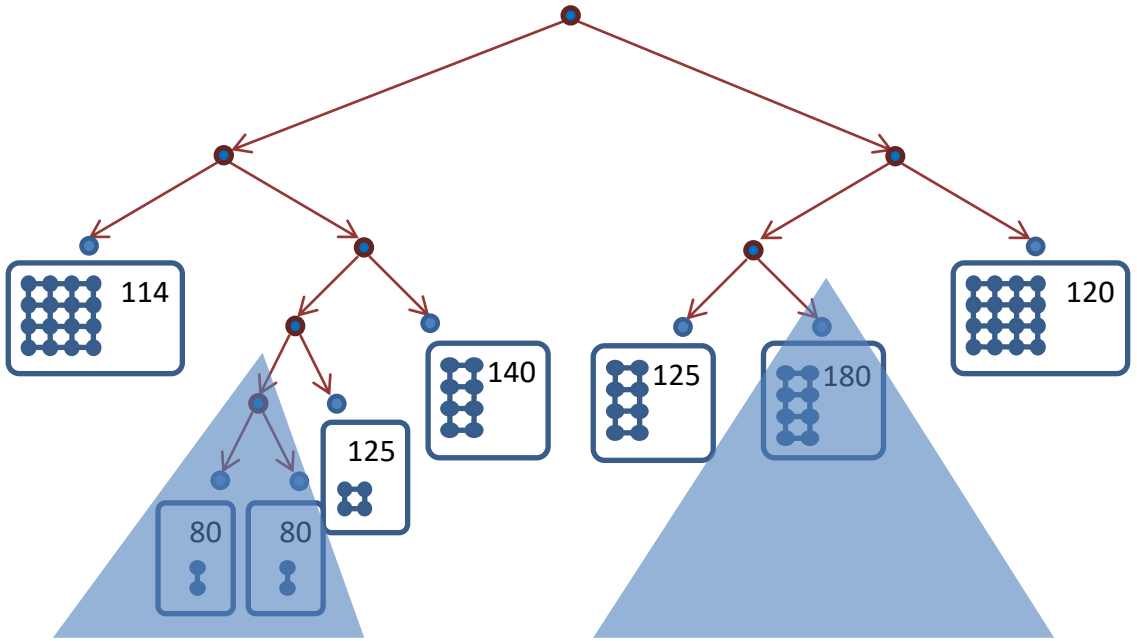
- Formally
  - A PBT **T** stores the current tiling as a full binary tree



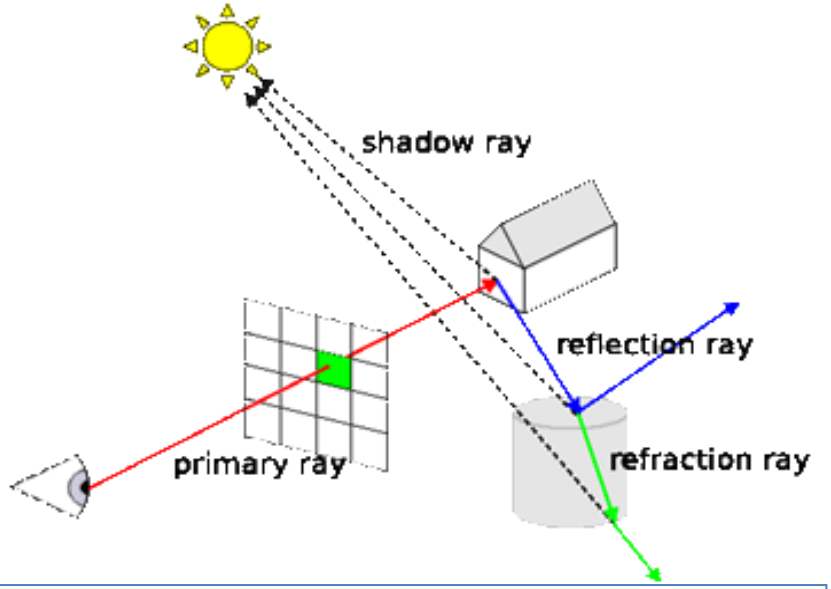


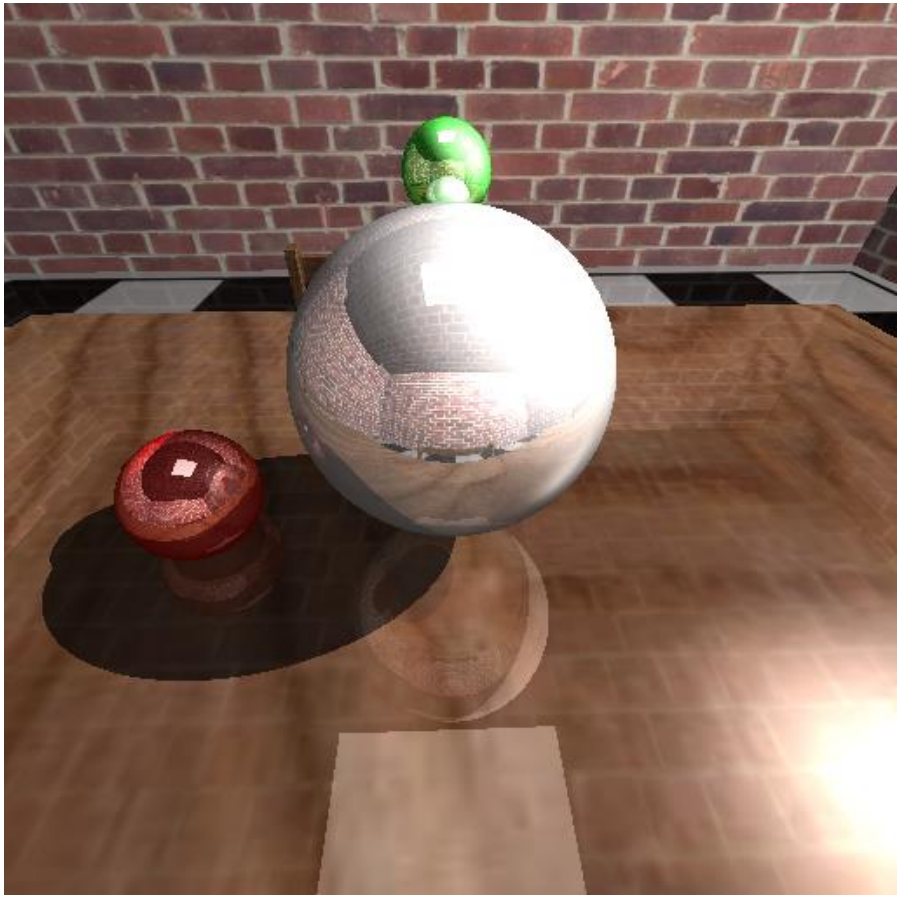
PBT Update





An example:
Ray Tracing Algorithm



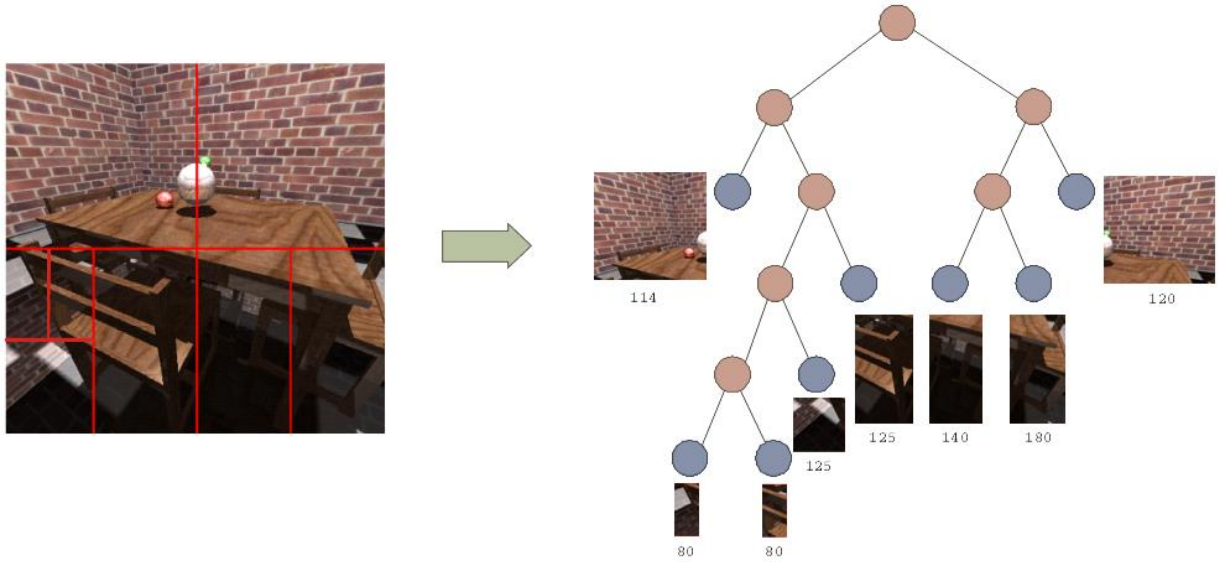


```
for each pixel
    for each sample
        ray object intersection
        lighting
        for each reflection/refraction rays direction
        integration for a sample
    integration for a pixel
```

loop

Exploiting PBT For PRT

- Frame => Step
- Pixel => item mesh
- At each step/frame, the mesh is partitioned into balanced tiles (rectangular area of pixels) and assigned to workers



PBT demo video

2. Load Balancing based on Cost Evaluation

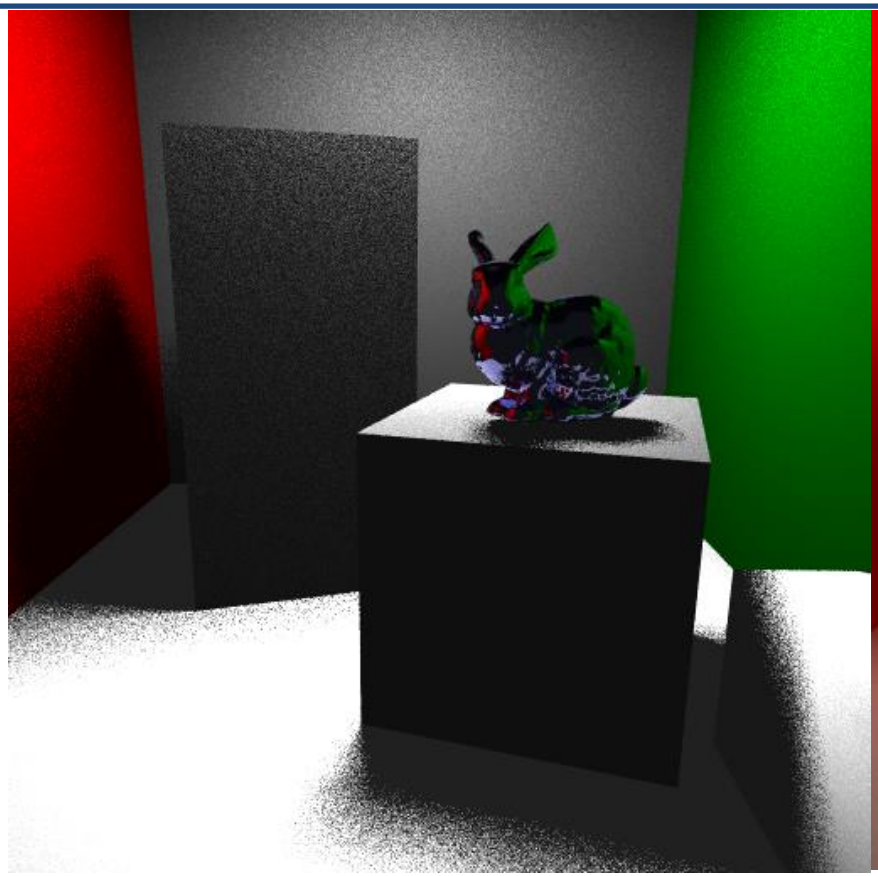
- Synergy Effects of Hybrid CPU-GPU Architectures for Interactive Parallel Ray Tracing

  GPU Cost Estimation for Load Balancing in Interactive Parallel Ray Tracing
  Biagio Cosenza, Ugo Erra, Carsten Dachsbacher
  (submitted to a journal)

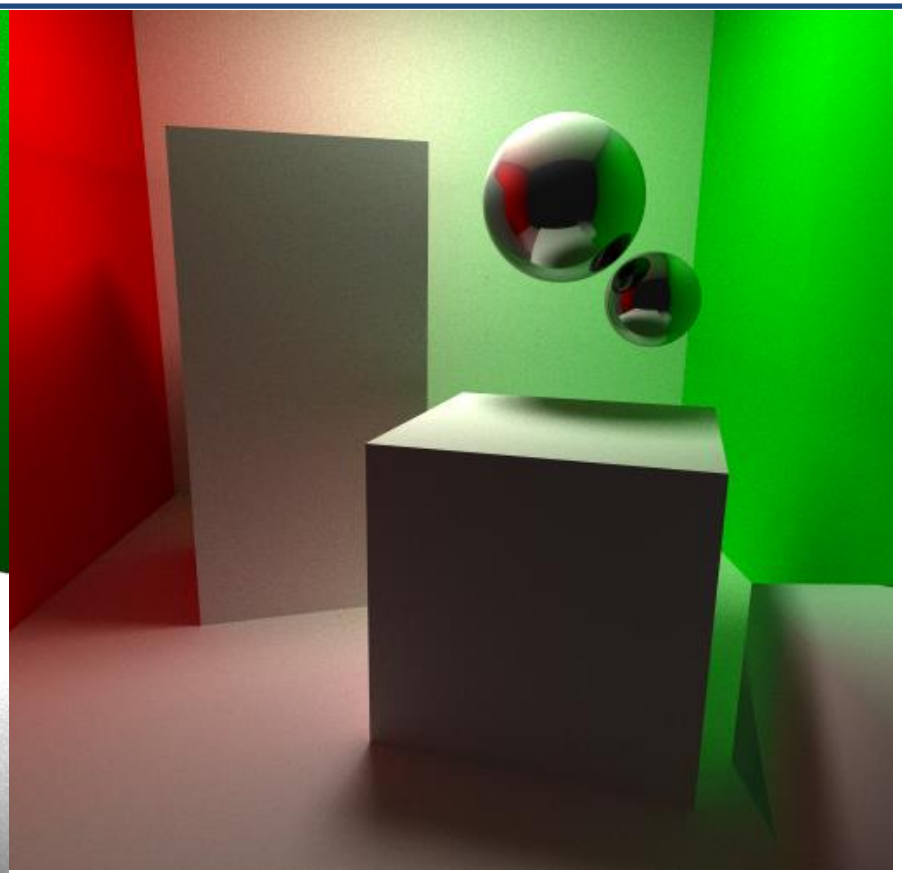
  Synergy Effects of Hybrid CPU-GPU architectures for Interactive Parallel Ray Tracing. *Biagio Cosenza*
  Science and Supercomputing in Europe, Research Highlights 2009.
  HPCEuropa2 Technical Reports ISBN 978-88-86037-23-5, CINECA, 2009.

- Project HPC-Europa2
  - Test ran at HLRS and VISUS (Universitaet Stuttgart), and ENEA Cresco Supercomputing

Ray Tracing Algorithms

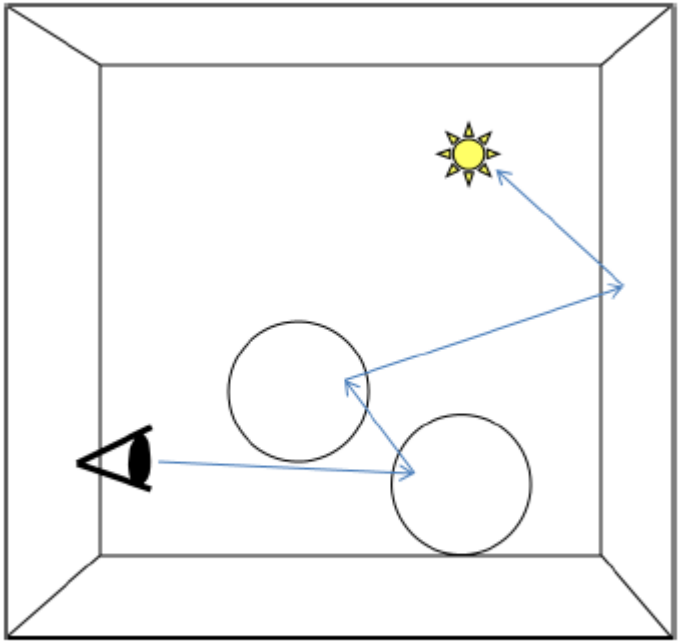


Whitted Ray Tracing, almost
4M of primary rays at 1024x1024

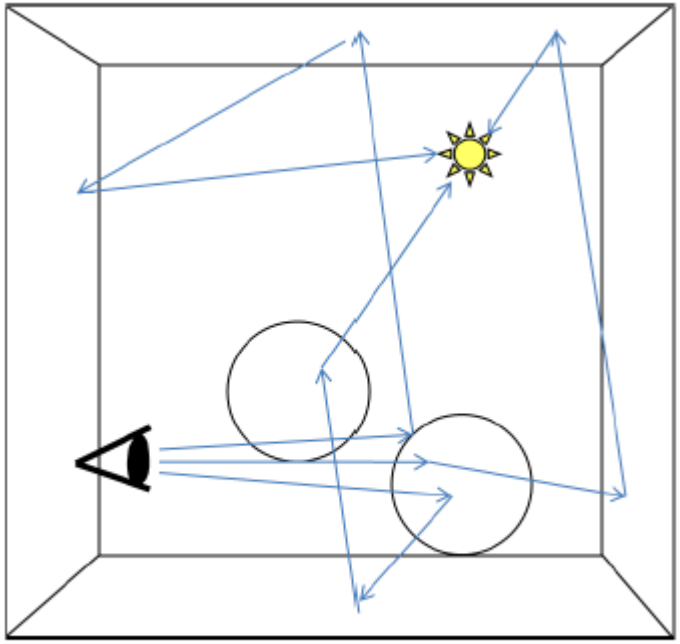


MC path tracing, more than
1000M of primary rays at 1024x1024

Ray Tracing Algorithms



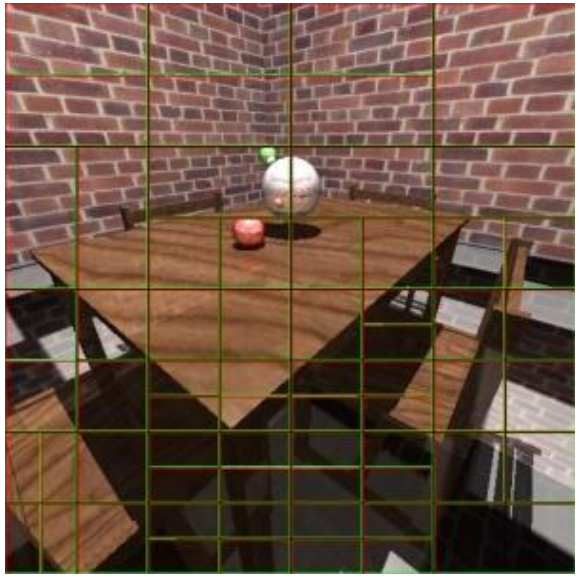
Whitted ray tracing



Kajila path tracing

Parallel Ray Tracing

- Image-space parallelization
- Master node = visualization node



Parallel Ray Tracing

- Our approach
  - Vectorial parallelism
    - Intel SSE instruction set
  - Multithreaded parallelism
    - pthread
  - Distributed memory parallelism
    - MPI
  - Use GPU Rasterization in order to speed up rendering (improving load balancing)

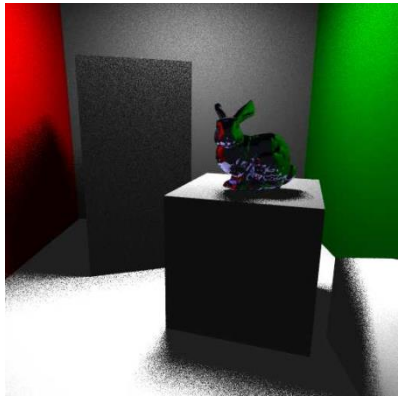
Parallel Ray Tracing

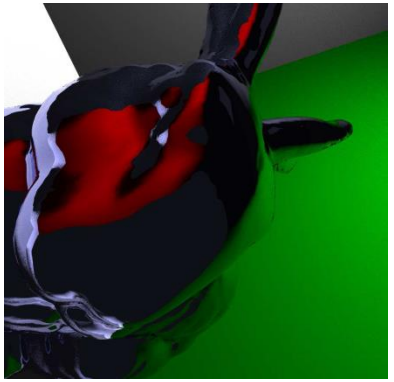
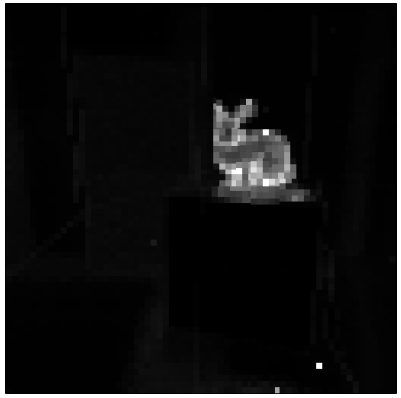
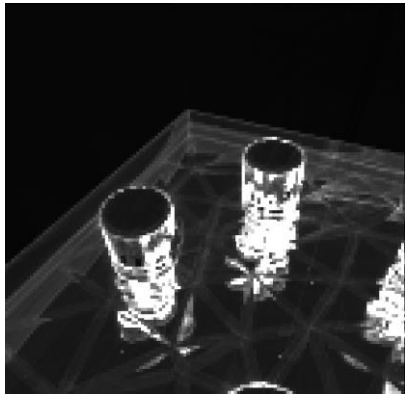
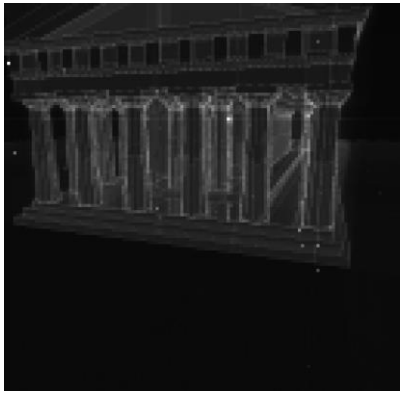
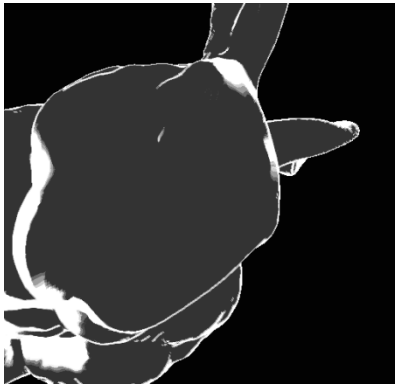
- Designed for cluster of workstations
  - Availability of multi-core processors and SIMD instructions
- Two task definition
  - Tile
  - Packet
- Synchrounous rendering
- Ray packets

Parallel Ray Tracing
Our approach to balance workload

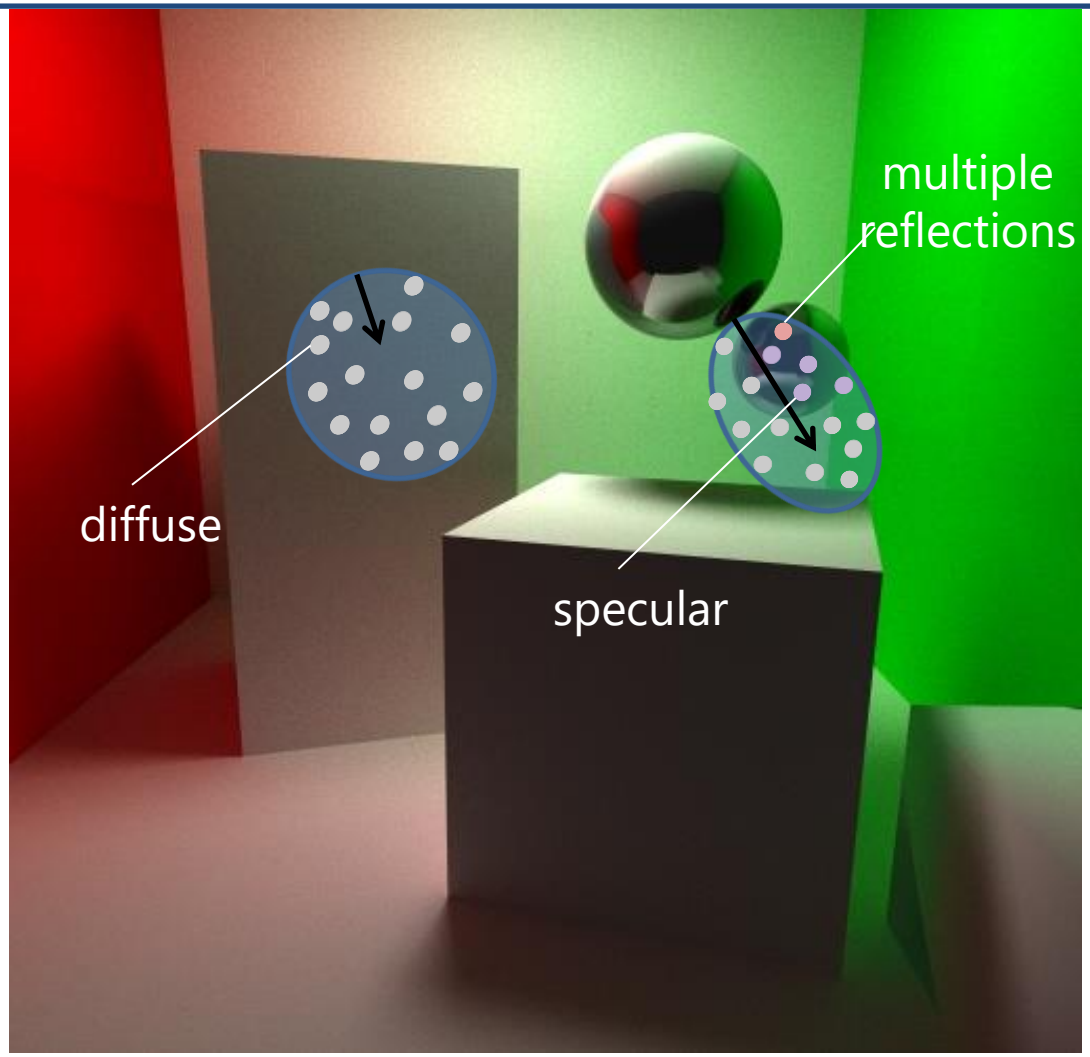
1. Compute a per-pixel, image-based estimate of the rendering cost, called *cost map*
2. Use the cost map for subdivision and/or scheduling in order to balance the load between workers
3. A dynamic load balancing scheme improves balancing after the initial tile assignment

Generate a Cost Map

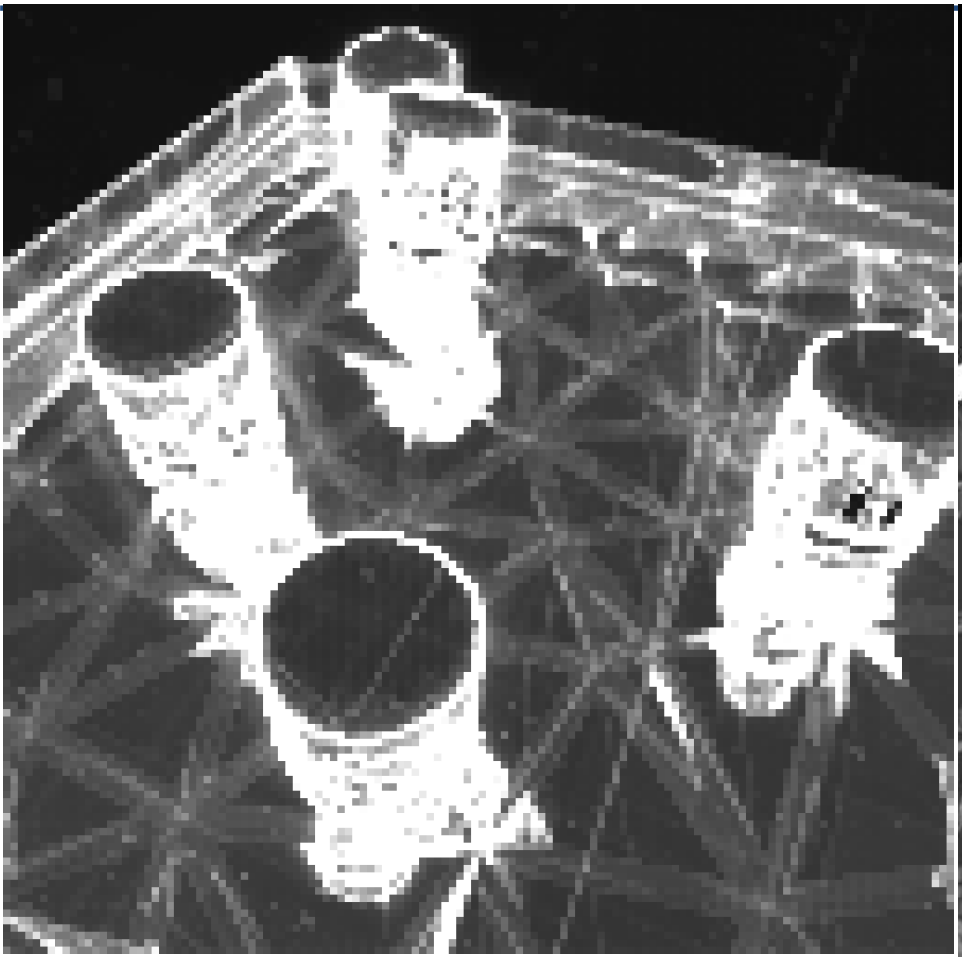


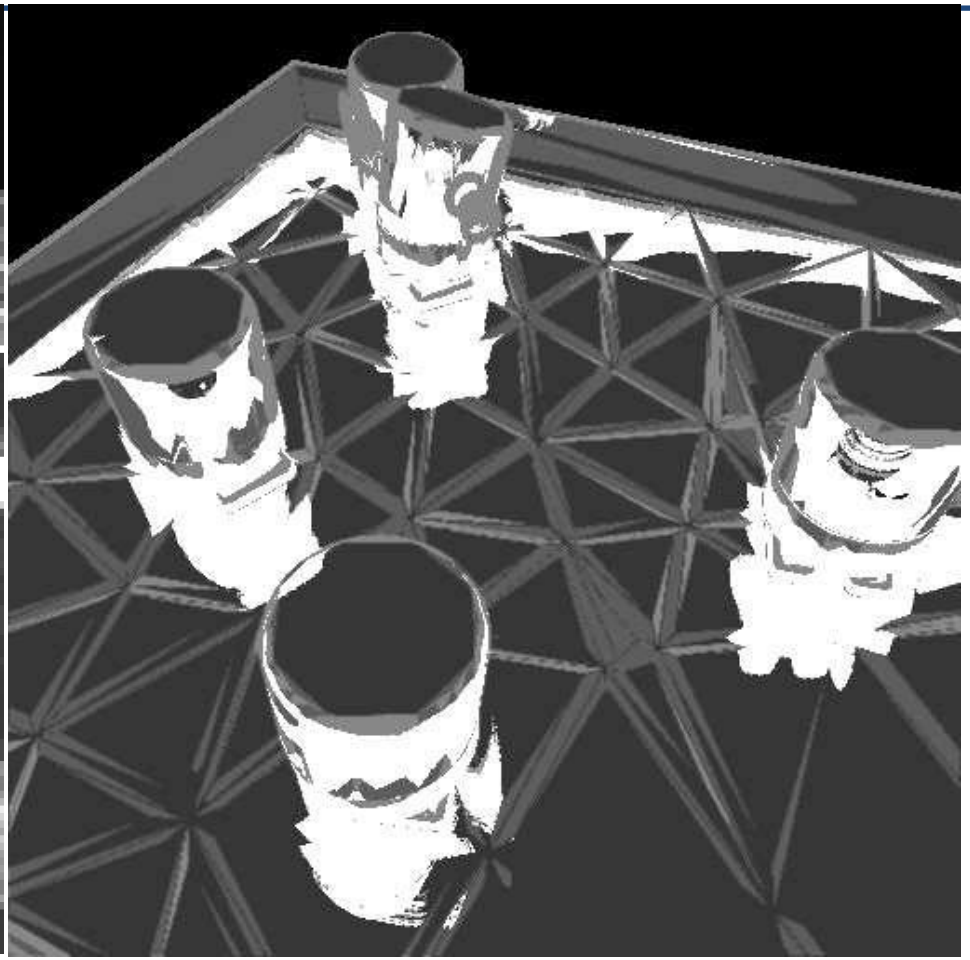







Generate a Cost Map



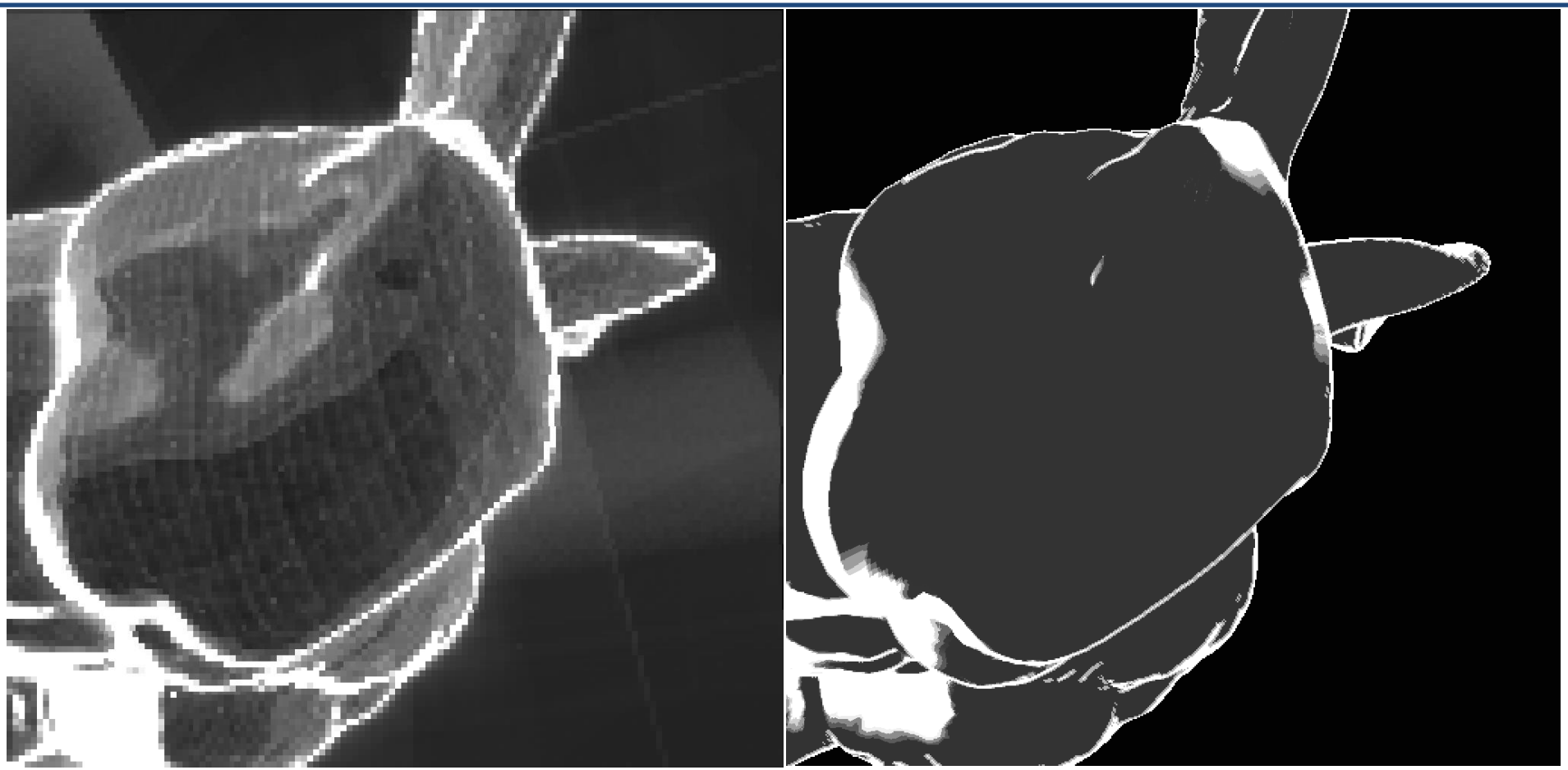
Cost Map Results





a comparison of the real cost (left) and
our GPU-based cost estimate (right)

Cost Map Results



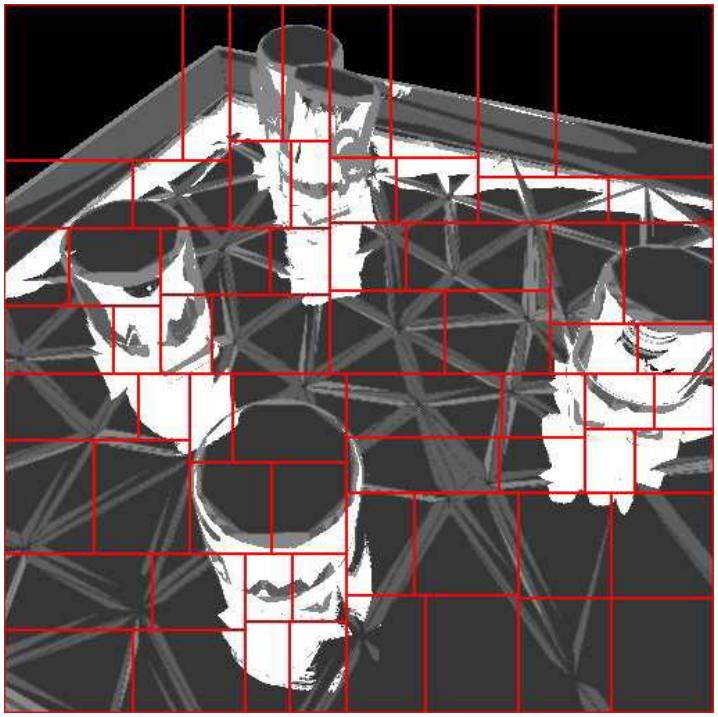
a comparison of the real cost (left) and
our GPU-based cost estimate (right)

Cost Map

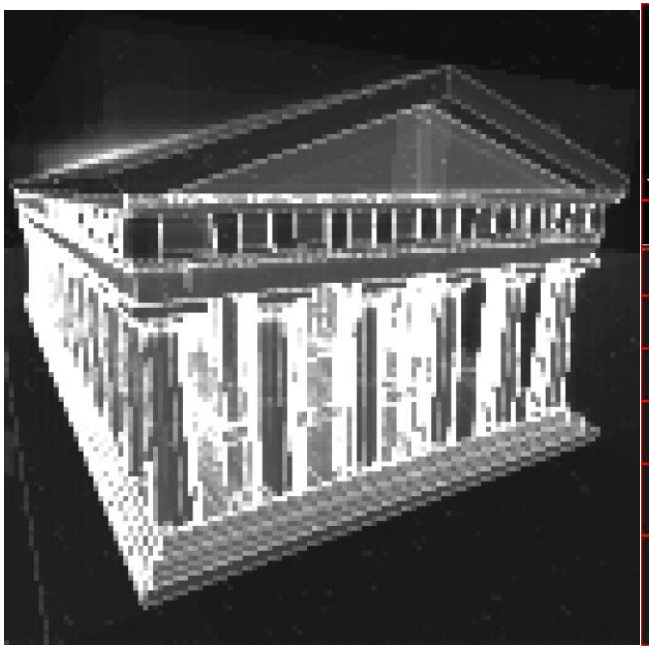
- Improves load balancing
- Two approaches
  - SAT Adaptive Tiling
  - SAT Sorting

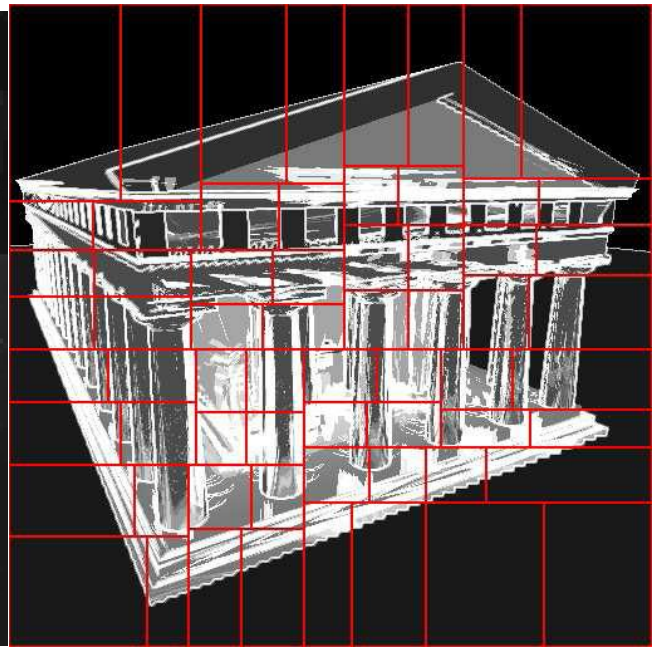
SAT Adaptive Tiling

- Adaptive subdivision of the image into tiles of roughly equal cost



Cost Map and Adaptive Tiling

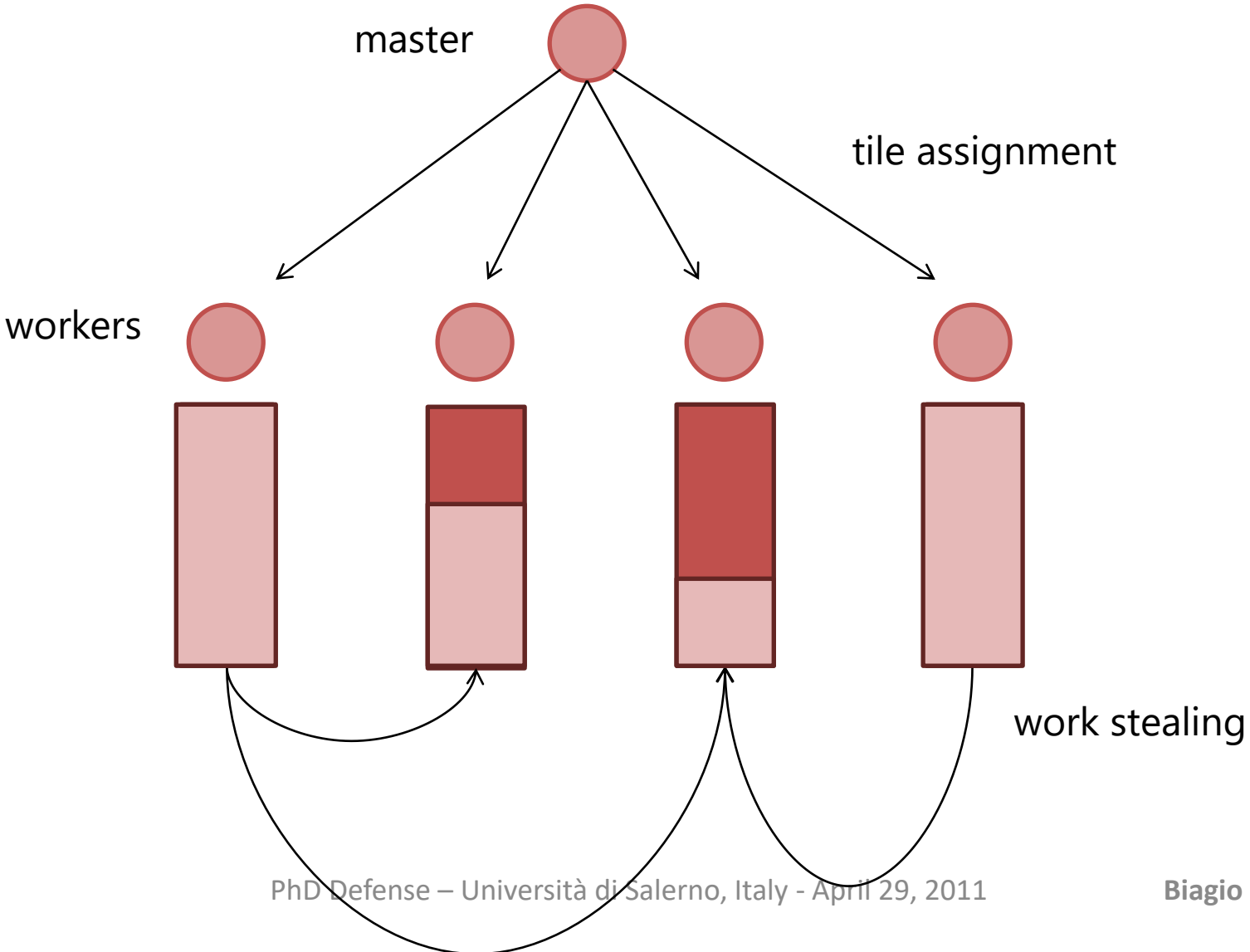




SAT Sorting

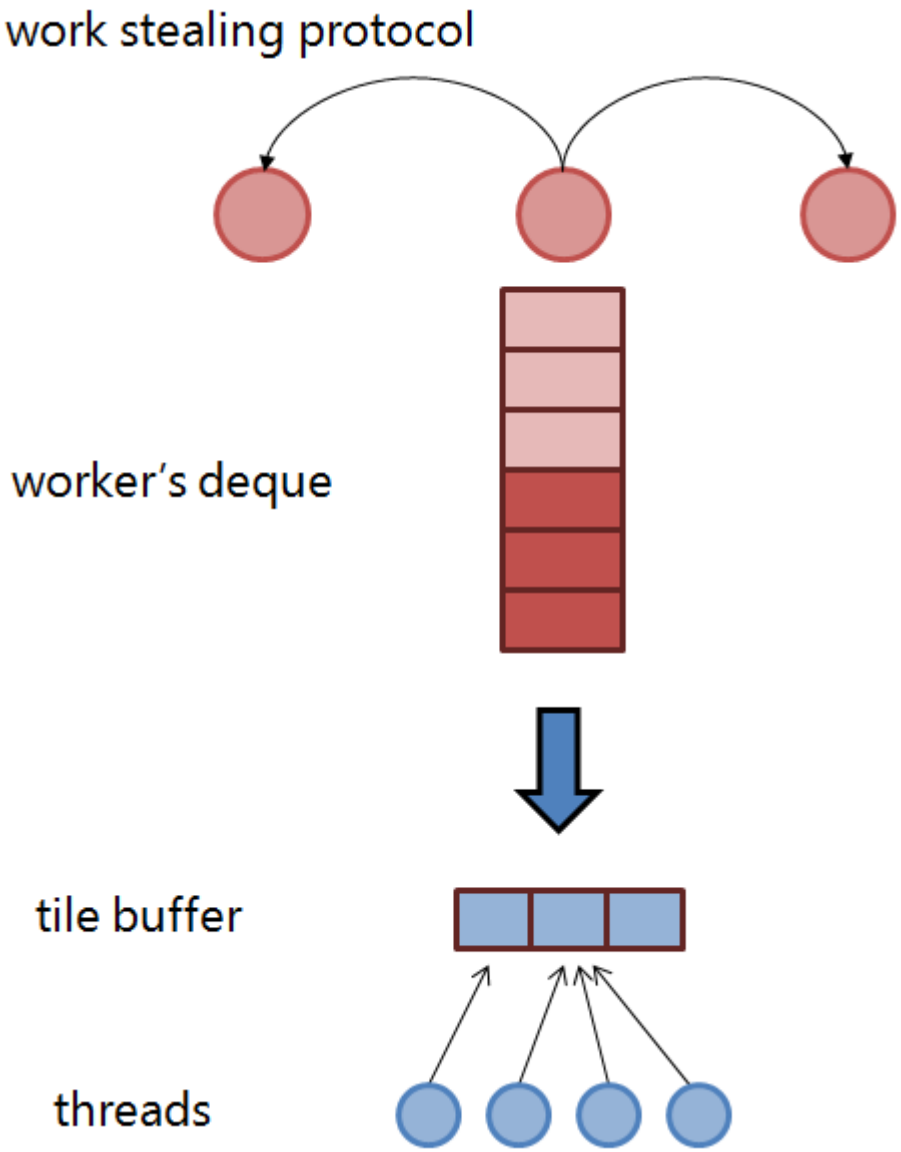
- Use a regular subdivision
- Sort tasks (=tiles) by estimate
- Dynamic load balancing
  - First schedule more expensive tasks
  - Cheaper ones at the end
  - Using work stealing, steals involve cheaper tasks
    - Assure a fine grained load balancing
- Use SAT (Summed Area Table) for fast cost evaluation of a tile
  - SAT [Crow & Franklin, SIGGRAPH ‘84]
  - SAT on the GPU [Hensley et al. 2005]

Work Stealing

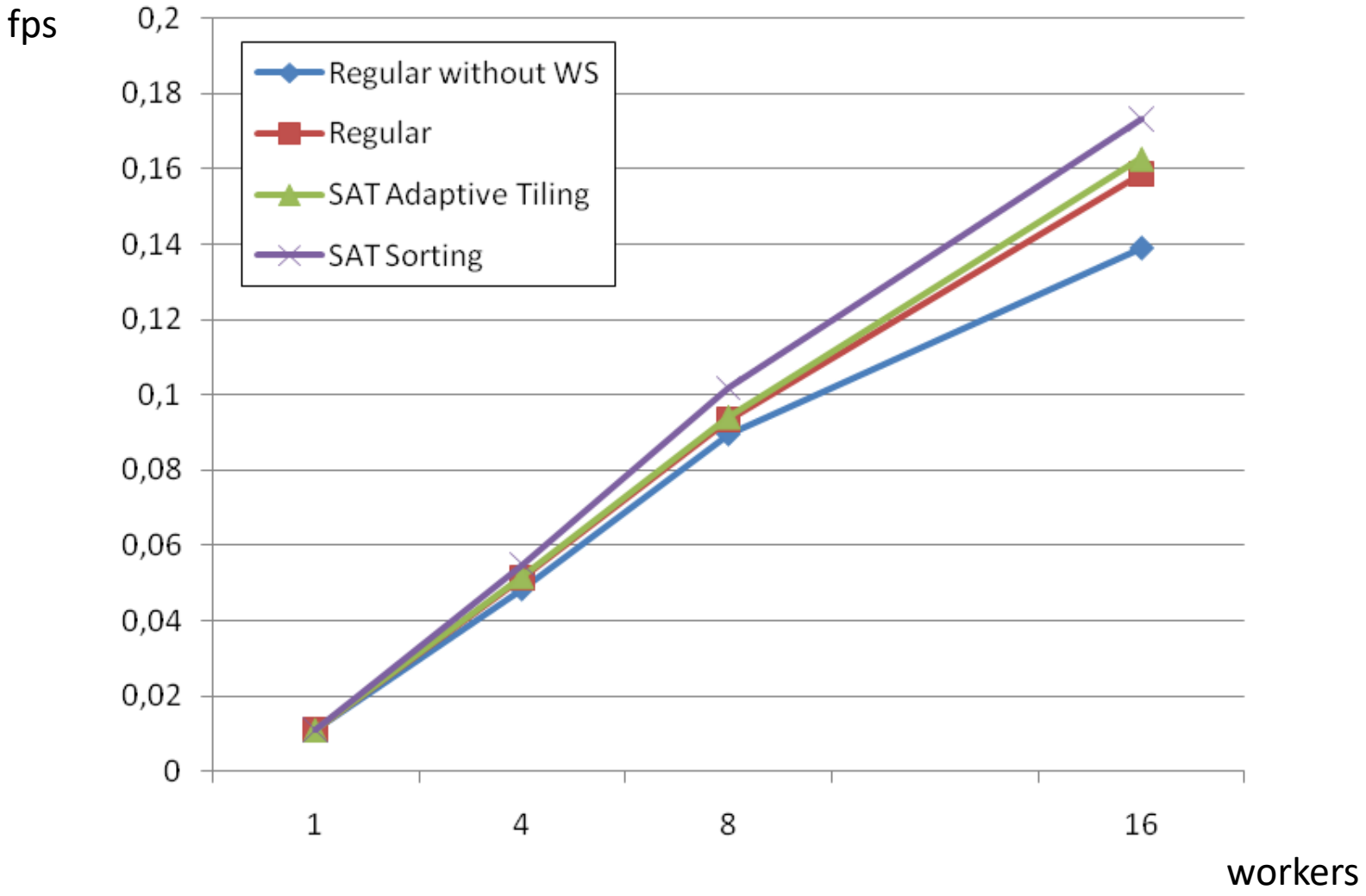


Packetization

- Two task definitions
  - **Tile** for parallel architecture (cluster-node level)
  - **Packet** for SIMD-based ray tracer (thread level)
- Approach 1-to-1
  - Problem to exploit best serial & parallel performances
- Approach 1-to-many
  - A tile is splitted in many packets
  - Decouple the two layers using the best granularities for both
    - 16 tiles, each one subdivided in packets of 8x8 rays


work stealing protocol
worker's deque
tile buffer
threads

Results


fps
0,2
0,18
0,16
0,14
0,12
0,1
0,08
0,06
0,04
0,02
0
Regular without WS
Regular
SAT Adaptive Tiling
SAT Sorting
1
4
8
16
workers

Notes on Cost Map-based Load Balancing

- SAT Adaptive Tiling
  - Needs a very accurate cost map estimate to have a gain

  "How much a tile is more expensive than another one?"
- SAT Sorting
  - It works also with a less accurate Cost Map

    "Which ones are the more expensive tiles?"
  - Well fit with the work stealing algorithm
  - Best performance

3. Agent-based Simulations

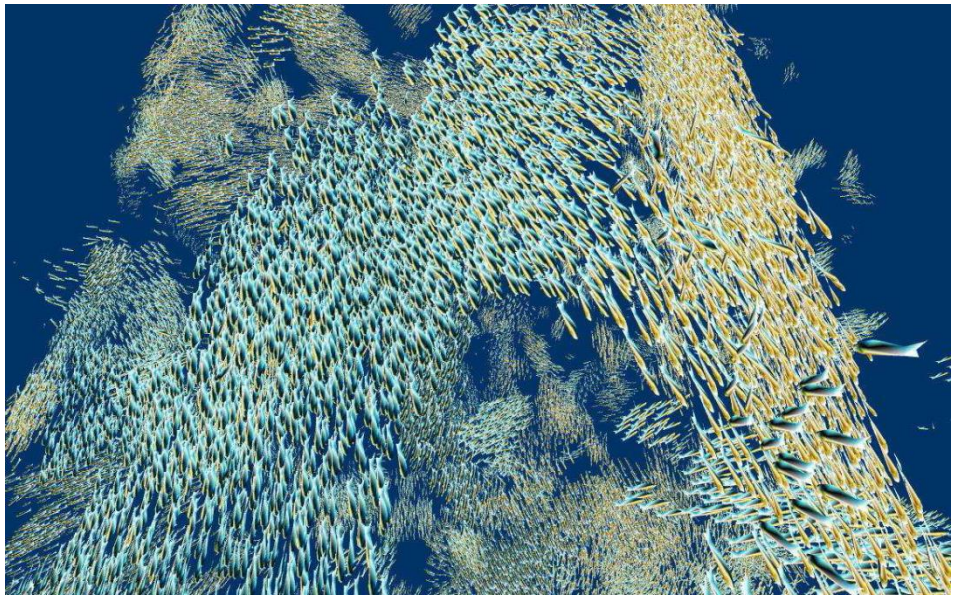
- Distributed Load Balancing for Parallel Agent-based Simulations

  Distributed Load Balancing for Parallel Agent-based Simulations
  Cosenza B., Cordasco G., De Chiara R., Scarano V.
  PDP 2011 - The 19th Euromicro International Conference on Parallel, Distributed and Network-Based Computing

- Project has been granted by a ISCRA Cineca

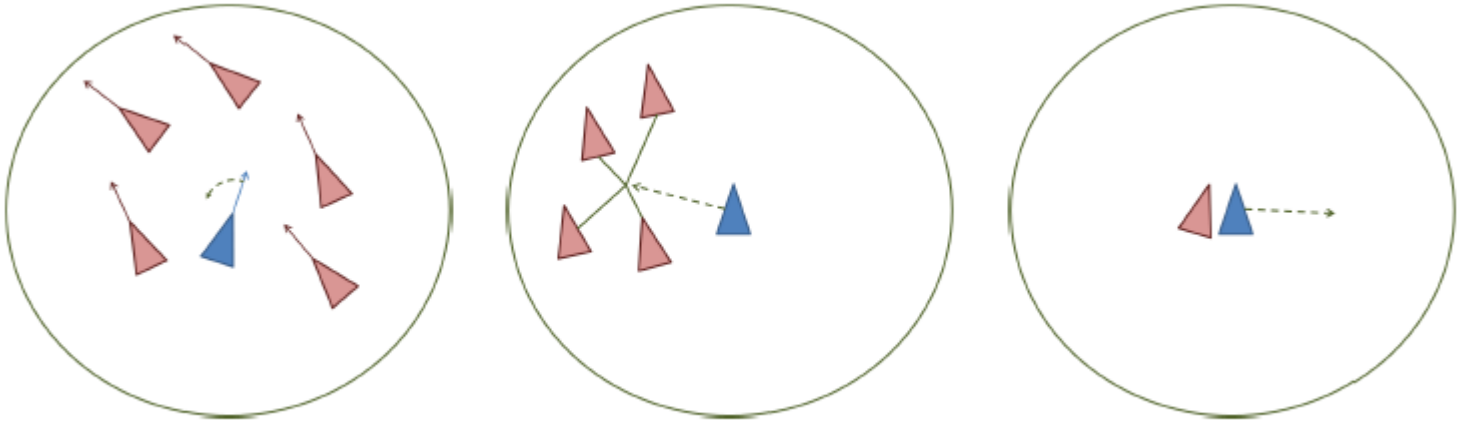
Agent-based Simulation

- Autonomous agents (individual), whose movements are related to social interactions among group members
- Examples
  - flock of birds
  - crowds
  - fish school
  - biological model



Reynolds Model

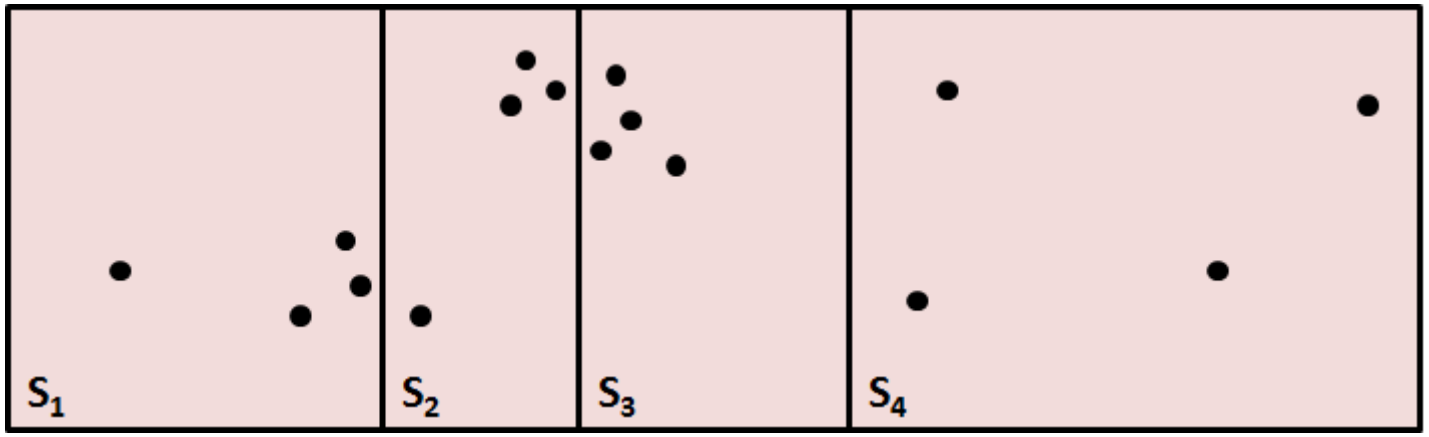
- Alignment, steer towards the average heading of local flock-mates
- Cohesion, steer to move toward the average position of local flock-mates
- Separation, steer to avoid crowding local flock-mates



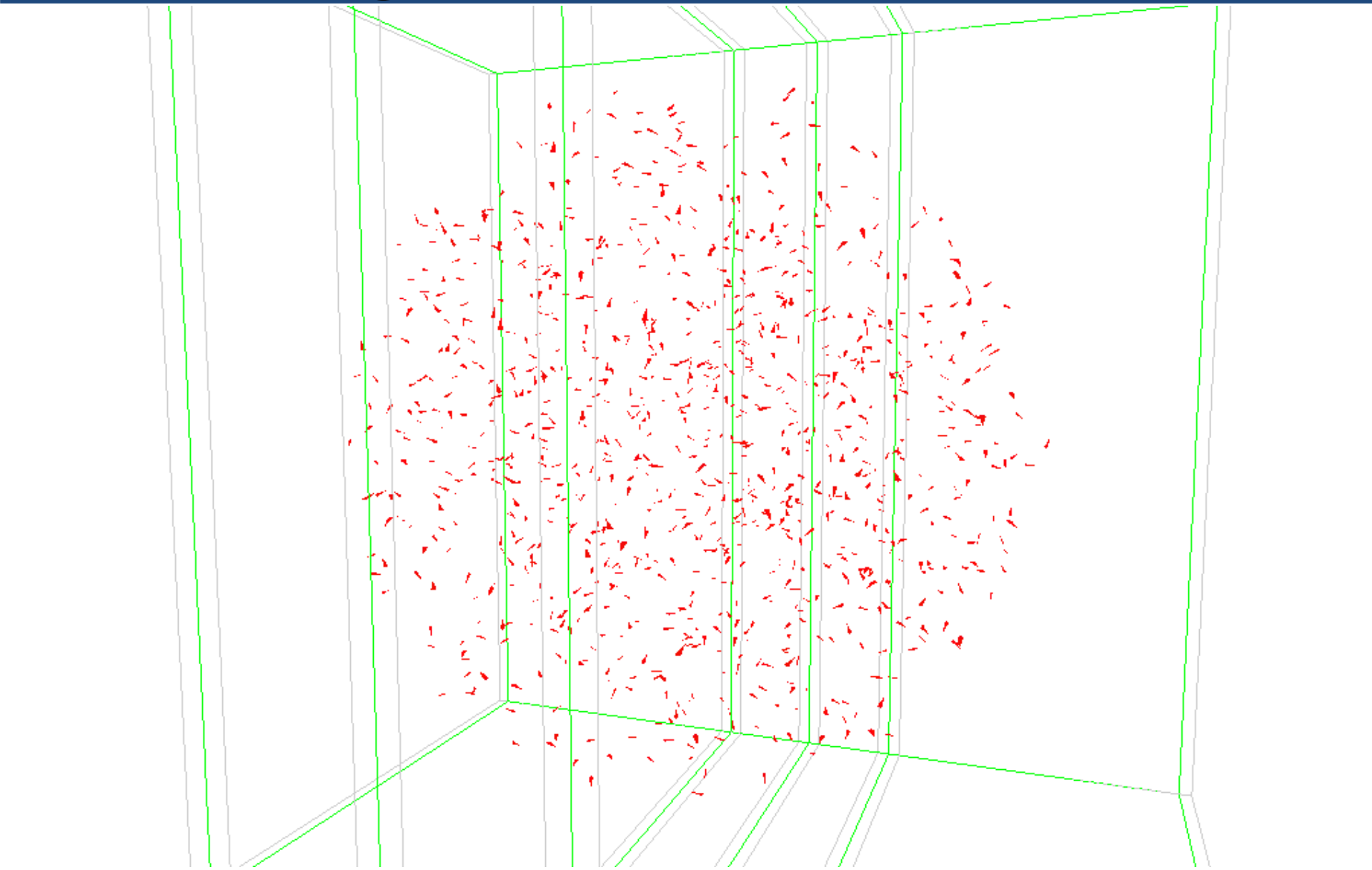
Alignment Cohesion Separation

Parallelization of Agent-based Simulation

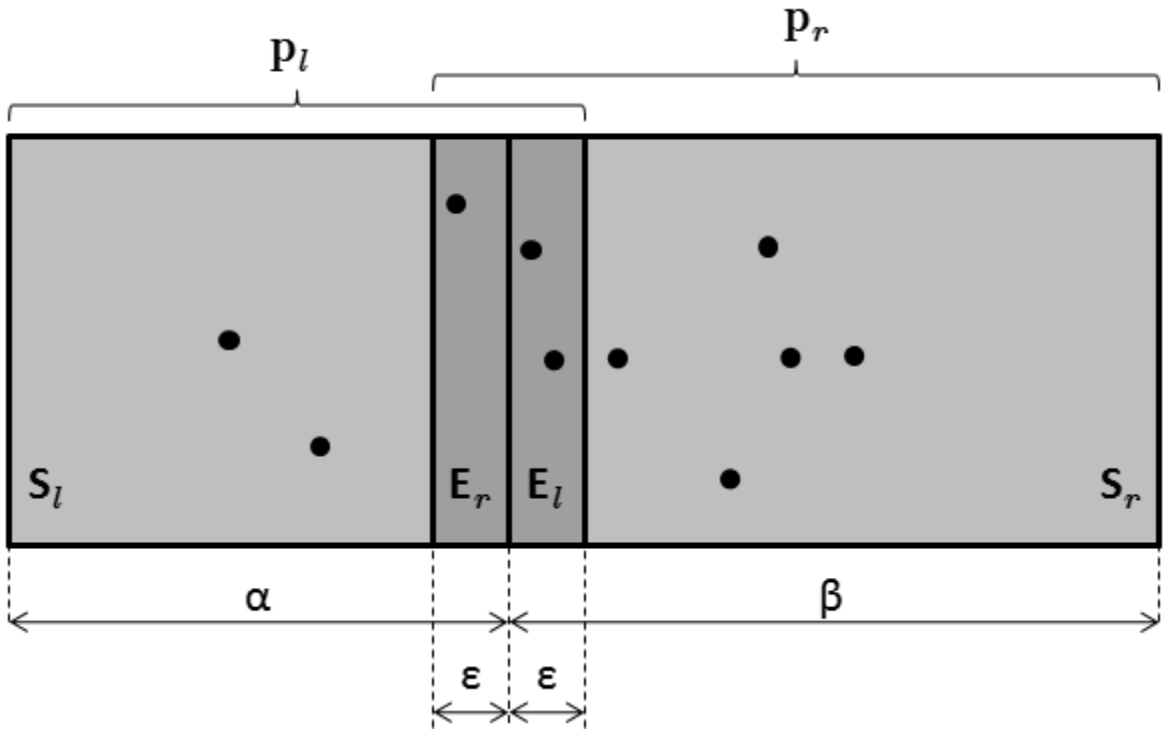
- Partitioning
  - Distributed (not centralized)
  - Even distribution (load balancing)



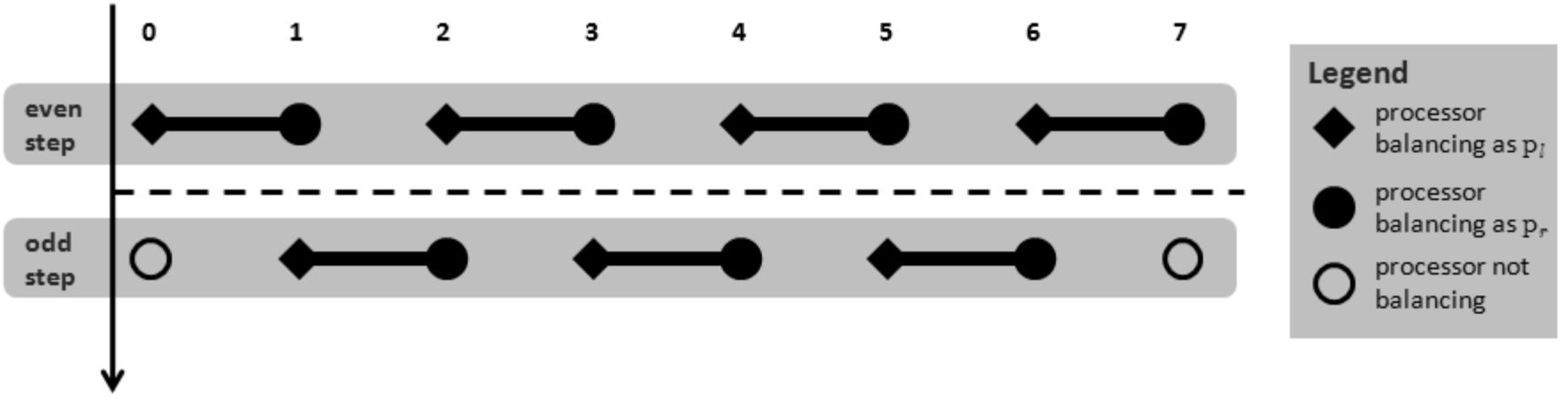
Parallelization of Agent-based Simulation



Two-procs schema

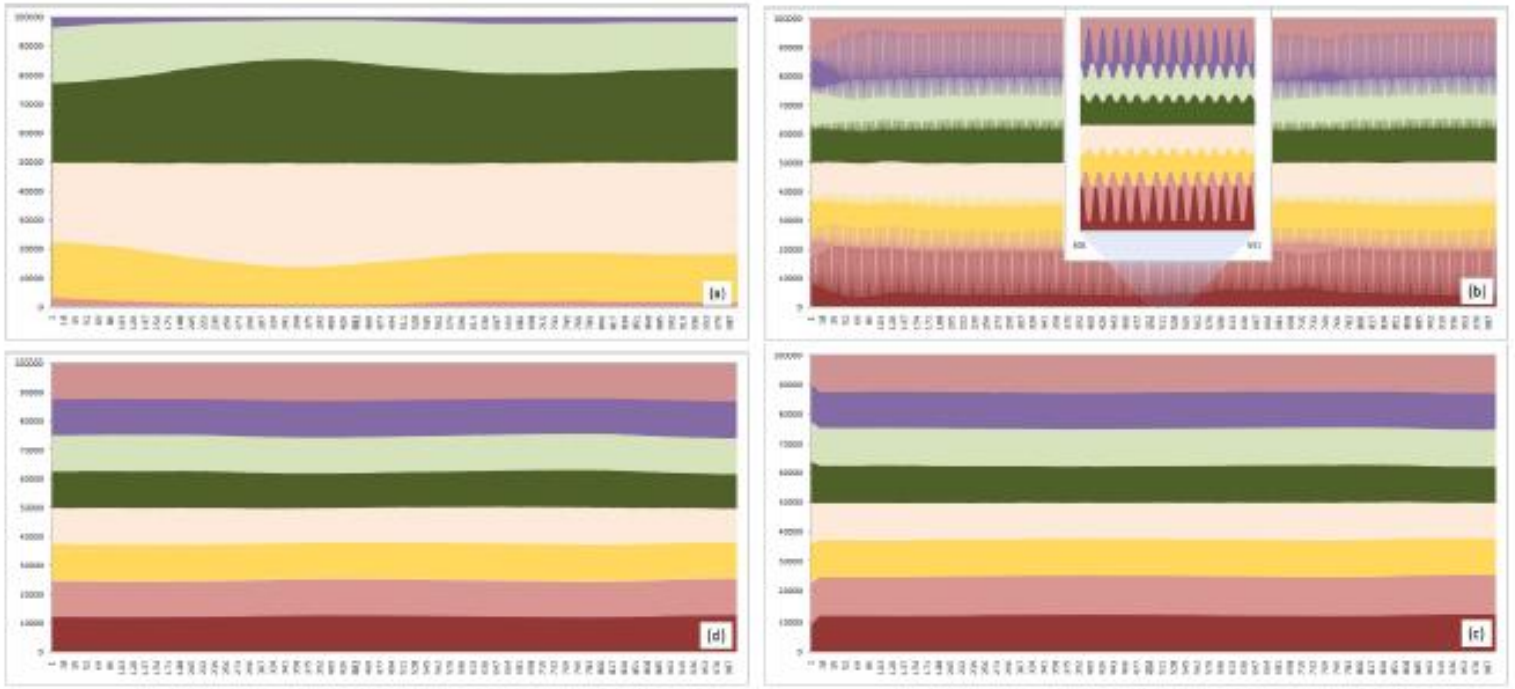


Multi-procs schema



Load distribution

- We proposed 4 different schemas
  - Exploiting geometrical properties of the agents' distribution



Agents Partitioning Demo Video

Conclusion

- Distributed vs Centralized
  - Fast vs Smart
  - Distributed scales with the number of processors
- Prediction/cost estimate-based approaches
  - Accuracy vs Computation time
- Work stealing
  - Sorting
  - Used on distributed system
- Hybrid architectures may offer non-trivial design

Publications (1)

- Experiences with Mesh-like computations using Prediction Binary Trees.
  Gennaro Cordasco, Biagio Cosenza, Rosario De Chiara, Ugo Erra, and Vittorio Scarano.
  Scalable Computing: Practice and Experience, Scientific International
  Journal for Parallel and Distributed Computing (SCPE), 10(2):173-187, June 2009.
- Load balancing in mesh-like computations using prediction binary trees.
  Biagio Cosenza, Gennaro Cordasco, Rosario De Chiara, Ugo Erra, and Vittorio Scarano.
  In ISPDC, pages 139-146, 2008.
- A Survey on Exploiting Grids for Ray Tracing. *Biagio Cosenza.*
  In Eurographics Italian Chapter Conference, pages 89-96, 2008.
- On Estimating the Effectiveness of Temporal and Spatial Coherence in Parallel Ray Tracing.
  Biagio Cosenza, Gennaro Cordasco, Rosario De Chiara, Ugo Erra, and Vittorio Scarano.
  In Eurographics Italian Chapter Conference, pages 97104, 2008.
- Load Balancing Techniques for Parallel Ray Tracing, 2008. *Biagio Cosenza.*
  Poster at HPC-Europa++ TAM-Workshop 2008, presented on 15-17/12/08 at HLRS
  Supercomputing Center, Universit¨at Stuttgart.

Publications (2)

- Synergy effects of hybrid CPU-GPU architectures for interactive parallel ray tracing. *Biagio Cosenza.*
  Science and Supercomputing in Europe, Research Highlights 2009. HPCEuropa2
  Technical Reports ISBN 978-88-86037-23-5, CINECA, 2009. ISBN 978-88-86037-23-5.
- Evaluation of adaptive subdivision schemas for parallel ray tracing.
  Biagio Cosenza.
  HPC-Europa: Science and Supercomputing in Europe, Technical Reports 2008 ISBN 978-88-86037-22-8, 2008. ISBN 978-88-86037-22-8.
- GPU Cost Estimation for Load Balancing in Parallel Ray Tracing.
  Biagio Cosenza, Carsten Dachsbacher, and Ugo Erra.
  (under review)
- Distributed load balancing for parallel agent-based simulations.
  Biagio Cosenza, Gennaro Cordasco, Rosario De Chiara, and Vittorio Scarano.
  In PDP2011 - 19th Euromicro International Conference on Parallel, Distributed and Network-Based Computing, 2011.

- Thanks for your attention
- Question?