Introduction: Ray Tracing Algorithms
Ray Tracing (RT) algorithms closely model physical light transport by shooting rays into the scene. In the simplest form, RT traces a ray for each pixel, from an eye or viewpoint through the pixel into the scene. Secondary rays can occur e.g. when rays hit reflective or refractive surfaces, for shadows, or for a more accurate simulation of the light transport in the scene, generating a huge computational cost.

Parallel Ray Tracing
Using complex shading drastically increases the number of rays, requiring a huge computational power. Improving performances to reach interactive rates requires to combine highly optimized RT implementations with massive amounts of computational power. Our system implements a screen-based Parallel RT: the image is subdivided into tiles distributed among workers, and the scene description is replicated. Each task, or tile, can be computed independently from every other task. Our implementation is tailored for distributed memory architectures.

The Load Balancing Problem
Rendering time of each pixel is influenced by a multitude of factors, such as shading algorithms, texture accesses, acceleration data structure queries. Uneven load balancing represents a bottleneck to the scalability. The general approach consists in subdividing the image in equally sized tiles, then using a balancing strategy during the scheduling. An important choice is the granularity of the subdivision of the images in tiles: a finer granularity, for instance, is good for balancing but needs a higher communication cost.

We consider a new way of managing load balancing based on adaptive subdivision: introducing a predictor unit, we are able to estimate the computation time needed by a tile and to arrange a “better tiling”.

Results
Our strategy provides a better scalability compared with a regular subdivision schema.

The optimal granularity for different number of processors needs a lower number of tiles, compared with a regular subdivision schema.

Moreover the overhead introduced by the adaptive subdivision schema is small compared with the gain in balancing tree and, hence, assures better performance.

Integrating this technique with known load balancing strategies is easy and effective. The technique can be extended to more general mesh-like computations.

References

Collaborations and credits
This is a joint work with G. Cordasco, R. De Chiara and V. Scarano of the D.I.A. at the Università degli Studi di Salerno. The author would like to thank Thomas Ehrl and Carsten Dachsbacher of the Institut für Visualisierung und Interaktive Systeme at the Stuttgart University for supporting the project and helpful discussions. Tests have been performed on a NEC Xeon E5647 cluster available at the HLR S High Performance Computing Center at the Stuttgart University.

Load Balancing Techniques for Parallel Ray Tracing
Biagio Cosenza
ISISLab, Dipartimento di Informatica ed Applicazioni
Università degli Studi di Salerno, Italy
email: cosenza@dia.unisa.it

The Load Balancing Problem
Rendering time of each pixel is influenced by a multitude of factors, such as shading algorithms, texture accesses, acceleration data structure queries. Uneven load balancing represents a bottleneck to the scalability. The general approach consists in subdividing the image in equally sized tiles, then using a balancing strategy during the scheduling. An important choice is the granularity of the subdivision of the images in tiles: a finer granularity, for instance, is good for balancing but needs a higher communication cost.

We consider a new way of managing load balancing based on adaptive subdivision: introducing a predictor unit, we are able to estimate the computation time needed by a tile and to arrange a “better tiling”.

Results
Our strategy provides a better scalability compared with a regular subdivision schema.

The optimal granularity for different number of processors needs a lower number of tiles, compared with a regular subdivision schema.

Moreover the overhead introduced by the adaptive subdivision schema is small compared with the gain in balancing tree and, hence, assures better performance.

Integrating this technique with known load balancing strategies is easy and effective. The technique can be extended to more general mesh-like computations.