Patch-Adaptive Relaxation in Multigrid Algorithms

Markus Kowarschik

Lehrstuhl für Informatik 10 (Systemsimulation)

Institut für Informatik

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Outline

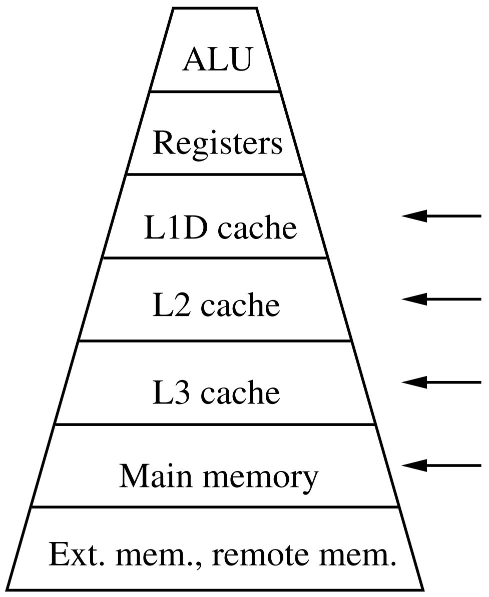
1. Introduction:
   - Memory hierarchies
   - Data locality optimizations
2. Adaptive relaxation (U. Rüde)
3. Patch-adaptive relaxation
4. Conclusions

Introduction: memory hierarchies

Motivation: CPU speed >> performance of DRAM-based memories

Approach: memory hierarchies, use of fast SRAM-based caches

Example: Intel Itanium 2 architecture (1,5 GHz, max. 6 GFLOPS):



Capacity	Bandwidth	Latency
16 KB	48 GB/s	1 cycle
256 KB	48 GB/s	5+ cycles
6 MB	48 GB/s	12+ cycles
2 GB	**6.4 GB/s**	**>150 cyc.**

$\Longrightarrow$ 1 memory access is more expensive than 600 FP operations!

Introduction: hierarchical memory architectures

Goal: Exploit the memory hierarchy as efficiently as possible!

Spectrum of **data locality optimizations** to enhance cache utilization:

- Hardware techniques; e.g., data prefetching
- Compiler-based techniques; e.g., elementary loop transformations
- Programming techniques; e.g.,
  - Data layout optimizations
  - Data access optimizations
- Inherently cache-aware algorithms;
  e.g., multigrid methods based on patch-adaptive relaxation

Programming techniques

Code transformations that can be automized (maintaining numerial properties)

1. **Data layout optimizations:**
   - Array padding: eliminate cache conflict misses
   - Array merging: cache-aware data structures to enhance spatial locality
   - Etc.

2. **Data access optimizations:**
   - Loop fusion: optimization of temporal locality
   - Loop blocking: optimization of spatial/temporal locality
   - Etc.

Previous work: `http://www10.informatik.uni-erlangen.de/dime`

Adaptive relaxation

Starting point: adaptive relaxation on $Ax = b$

Assumption: $A = (a_{i,j}) \in \mathbf{R}^{n \times n}$, large, sparse, symmetric positive definite

$$Ax = b \quad \Longleftrightarrow \quad \frac{1}{2} x^T A x - x^T b = \min!$$

Scaled residual: $\theta_i(x) := a_{i,i}^{-1} e_i^T (b - Ax)$

Motivation:
Error reduction for one elementary relaxation step $x \leftarrow x + \theta_i(x) e_i$:

$$||x^{\text{old}} - x^*||_E^2 - ||x^{\text{new}} - x^*||_E^2 = a_{i,i} \theta_i(x^{\text{old}})^2$$

x^*: exact solution

Adaptive relaxation

Definition of an adaptive processing strategy:
"Relax, where the error is reduced most efficiently (i.e., where θ_i is large)"

Required: active set: set of indices of nodes with "large" scaled residuals

```
Algorithm: adaptive relaxation
1: while ActiveSet $\neq 0$ do
2:     pick $i \in$ ActiveSet
3:     ActiveSet $\leftarrow$ ActiveSet $\backslash \{i\}$
4:     if $|\theta_i(x)| > \theta$ then
5:         $x \leftarrow x + \theta_i(x) e_i$
6:         ActiveSet $\leftarrow$ ActiveSet $\cup$ Neighbors($i$)
7:     end if
8: end while
```

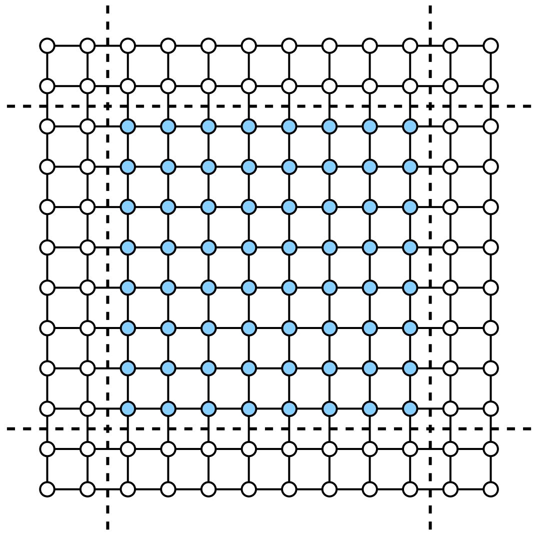
Algorithm terminates as soon as active set is empty; i.e., $\forall i : \theta_i \leq \theta$

Fully adaptive multigrid:
Adaptive relaxation on the entire multigrid hierarchy (U. Rüde, M. Griebel)

Patch-adaptive relaxation

Data locality optimization: Exploit the flexible update strategy!

Patch adaptivity: patch-based processing (instead of node-based processing)



Consequences:

$\Longrightarrow$ Reduced overhead through reduced granularity

$\Longrightarrow$ Cache-efficient if

- patch size is chosen appropriately and
- patches are relaxed consecutively

Patch-adaptive relaxation

Multigrid setting:

Patch-adaptive relaxation scheme to be used as smoother

$\Longrightarrow$ Efficient uniform error reduction

Preliminary experimental results:

- Increased robustness in the case of singularities through **local relaxation** (A. Brandt)
- Enhanced reuse of cache contents:
  - First simulation results on Intel P4 architecture:
    L2 cache miss rate decreases by more than 60%
  - Performance tuning: work in progress ... (Time to solution matters!)

Conclusions

- Performance results for cache-aware programming techniques have been reported on previously (e.g., PARA'02)
- Focus: inherently cache-conscious numerical algorithms
- Particularly: design/implementation of a patch-adaptive multigrid smoother
  - Based on the fully adaptive multigrid method
  - First numerical/performance results are promising
- Future trends:
  - CPU-DRAM gap will continue to increase $\Longrightarrow$ need for locality
  - Cache sizes will continue to increase (2005: Intel Itanium2 dual-core: 24 MB L3 cache on-chip)
- See `http://www10.informatik.uni-erlangen.de/dime`