Shoal: self-tuning for memory allocation and access

Stefan Kaestle, Reto Achermann
4th year PhD student
Advisor: Timothy Roscoe

Systems Group @ ETH Zurich
Observation: HW complex

• Multicores: complex memory subsystem
  – non-uniform memory access (NUMA)
  – complex interconnect network
  – hardware features
    • different page sizes (4KB, 2/4MB, 2GB)
    • DMA engines

  – no single global address space
  – no cache-coherence

worse:
diversity, constant changes
Implications / Problem

→ programmers struggle

• where to allocate memory?
• how to access this memory?

if wrong → no scalability & bad performance:
  – interconnect contention
  – saturation of memory controllers

Stefan Kaestle <stefan.kaestle@inf.ethz.ch>
Goal

• **simplify** the life of programmers
• **hardware-aware** memory allocation & access
• automatic tuning, **without**:
  - ✗ constantly fine-tune applications
  - ✗ understand the hardware
• support for wide range of current/future HW

Stefan Kaestle <stefan.kaestle@inf.ethz.ch>
Our System: **Shoal**

- **array** abstraction
- **rich alloc** call: **access patterns** as arguments
  - manually or automatic from high-level description
- various implementations
  - e.g. replicated, distributed, partitioned
- **selection automatically** based on *a)* access patterns and *b)* HW characteristics
Overview

- low-level code (e.g. C)
  - Shoal array abstraction
- hardware characteristics
- access patterns
- program
  - Shoal library
Overview

- high-level program (e.g. Green Marl)
- high-level compiler
- low-level code (e.g. C)
- Shoal array abstraction
- hardware characteristics
- access patterns
- program
- Shoal library

Shoal library
Procedure pagerank(G: Graph, e,d: Double, max: Int; pg_rank: Node_Prop<Double>)
{
    Double diff;
    Int cnt = 0;
    Double N = G.NumNodes();
    G.pg_rank = 1 / N;
    Do {
        diff = 0.0;
        Foreach (t: G.Nodes) {
            Double val = (1-d) / N + d * \text{Sum}(w: t.InNbrs)\{w.pg_rank / w.OutDegree()\} ;
            diff += | val - t.pg_rank |;
            t.pg_rank <= val @ t;
        }
        cnt++;
    } While ((diff > e) && (cnt < max));
}
Evaluation: Scalability

Green Marl

- pagerank
- hop_dist
- triangle_count

OpenMP scales fairly well
Evaluation: Scalability

Green Marl

**pagerank**

Runtime (x10^5 [ms])

- **OpenMP**
- **Shoal**

**hop_dist**

Runtime (x10^5 [ms])

- **OpenMP**
- **Shoal**

**triangle_count**

Runtime (x10^5 [ms])

- **OpenMP**
- **Shoal**

*up to 2x improvement*

Stefan Kaestle <skaestle@inf.ethz.ch>
Conclusion / Future work

• library to **auto-tune** memory allocation & access
  – based on: **access patterns & hardware**
• 2x improvement for Green Marl graph algorithms
  – **no modifications** to input program

• Next:
  – scheduling
  – synchronization
  – heterogeneous machines
  ➔ long term: maybe multikernel programming model

Stefan Kästle <stefan.kaestle@inf.ethz.ch>