A Scalable Approach to Thread-level Speculation

Steffan et. al. at CMU
Goals

- Automatic, dynamic parallelization
- Scalability is key: they want to run on SMT/SMP/CMP
Execution Model

- Spawn a bunch of threads
  - One per loop iteration is a good idea
  - Other granularities are possible as well.
- Register dependences are handled by the compiler/programmer. It’s static, so it’s easy.
- “Just” get the memory right.
(a) Example pseudo-code
while(continue condition) {
    ...
    x = hash[index1];
    ...
    hash[index2] = y;
    ...
}

(b) Execution using thread-level speculation

Figure 1. Example of thread-level speculation.
Architecture

(a) General architecture

Physically Private Caches

Physically Shared Caches

Processor Actions

External Actions

Interconnection Network
How they do it

- Hijack the coherence protocol
- A common trick nowadays
- It’s the obvious place to track sharing
- It’s sort of slow already, so extra overhead is not such a big deal.
- Only works with private caches.
- Some shared cache rules for SMT machines.
Hacking MESI

- New states track speculative accesses
- If an older thread writes to something a younger thread has read or written, there’s a mis-speculation.
Making progress

• The oldest thread is “home free”
  • It’s modifications are no longer speculative
• All cache lines must transition out of speculative states
• They keep a table of the affected cache lines.
Performance

(a) Execution Time
Performance
In Context

• Speculative threading has a looong history

• Lots and lots of papers

• It’s part of a larger field called “non-traditional parallelism”

• How do you use many CPUs to speed up one thread.

• Dean Tullsen does a lot of stuff like this.

• This also sounds a lot like transactional memory.
Your Questions

- How far can this non-traditional parallelism get us?