TAPE: a Transactional Application Profiling Environment

Hassan Chafi, Chi Cao Minh, Austen McDonald, Brian D. Carlstrom,
JaeWoong Chung, Lance Hammond,
Christos Kozyrakis, and Kunle Olukotun

Computer Systems Laboratory
Stanford University

http://tcc.stanford.edu

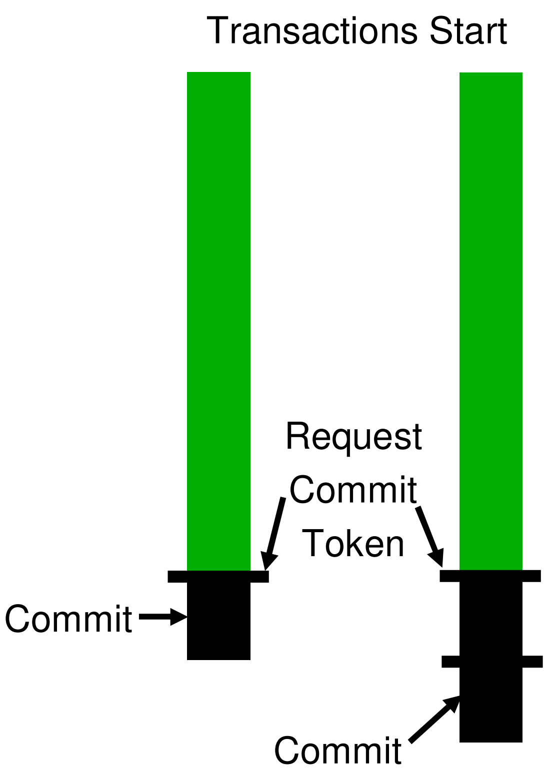
Optimizing Parallel Performance

- CMPs are here but parallel programming is still difficult
  - Need correct and fast parallel executables
- Transactional memory simplifies correct parallel programming
  - No locks
  - Speculative parallelization
- The Issue is now performance tuning

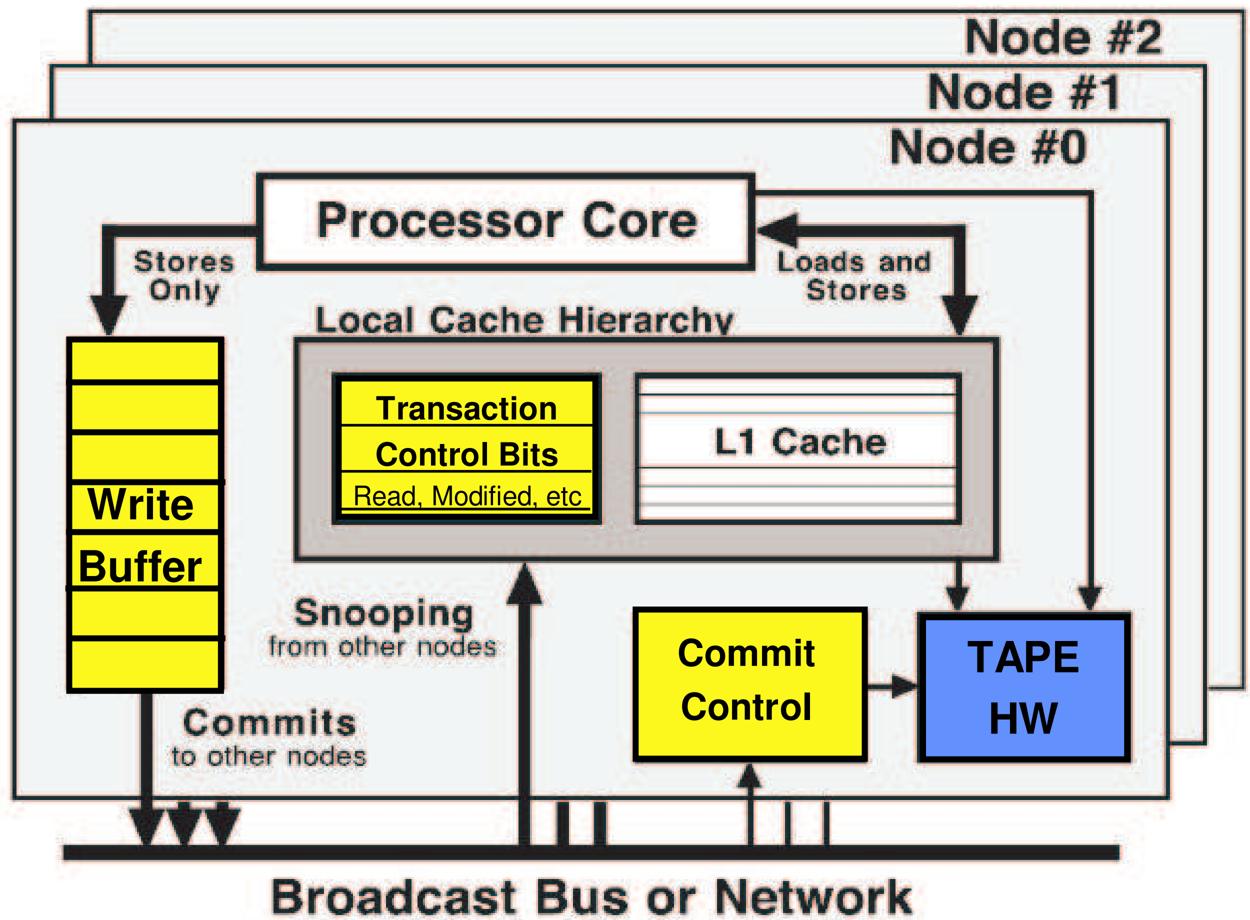
- TAPE: a system for performance profiling of transactional applications
  - Expressive: tracks all performance bottlenecks
  - Accurate: identifies bottleneck location in source code
  - Easy to use: leads to optimal performance in few tuning steps
  - Low overhead: negligible area & performance cost

- TAPE allows for continuous profiling, even on production runs

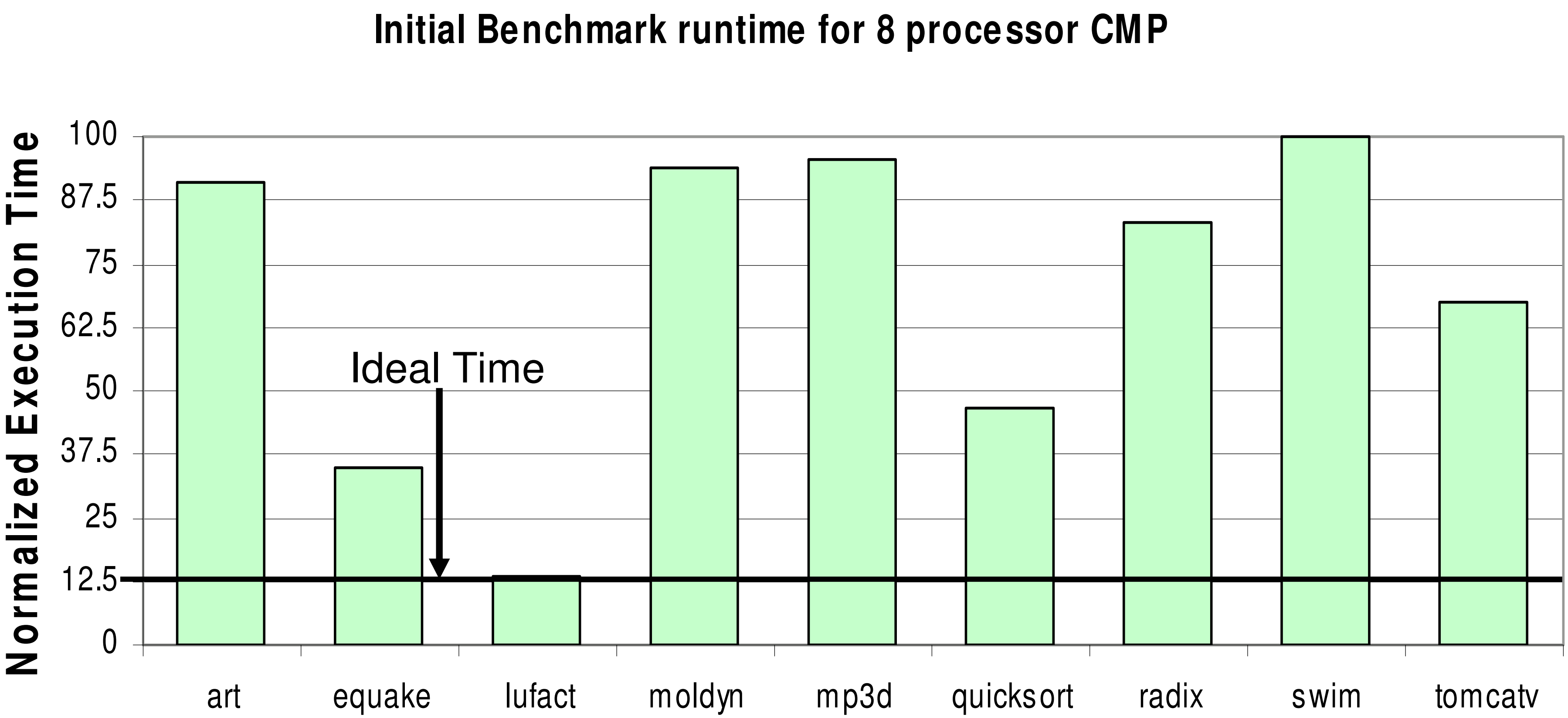
TCC Architecture for Transactional Execution



Transaction Timeline



Out-of-the-box TCC Performance

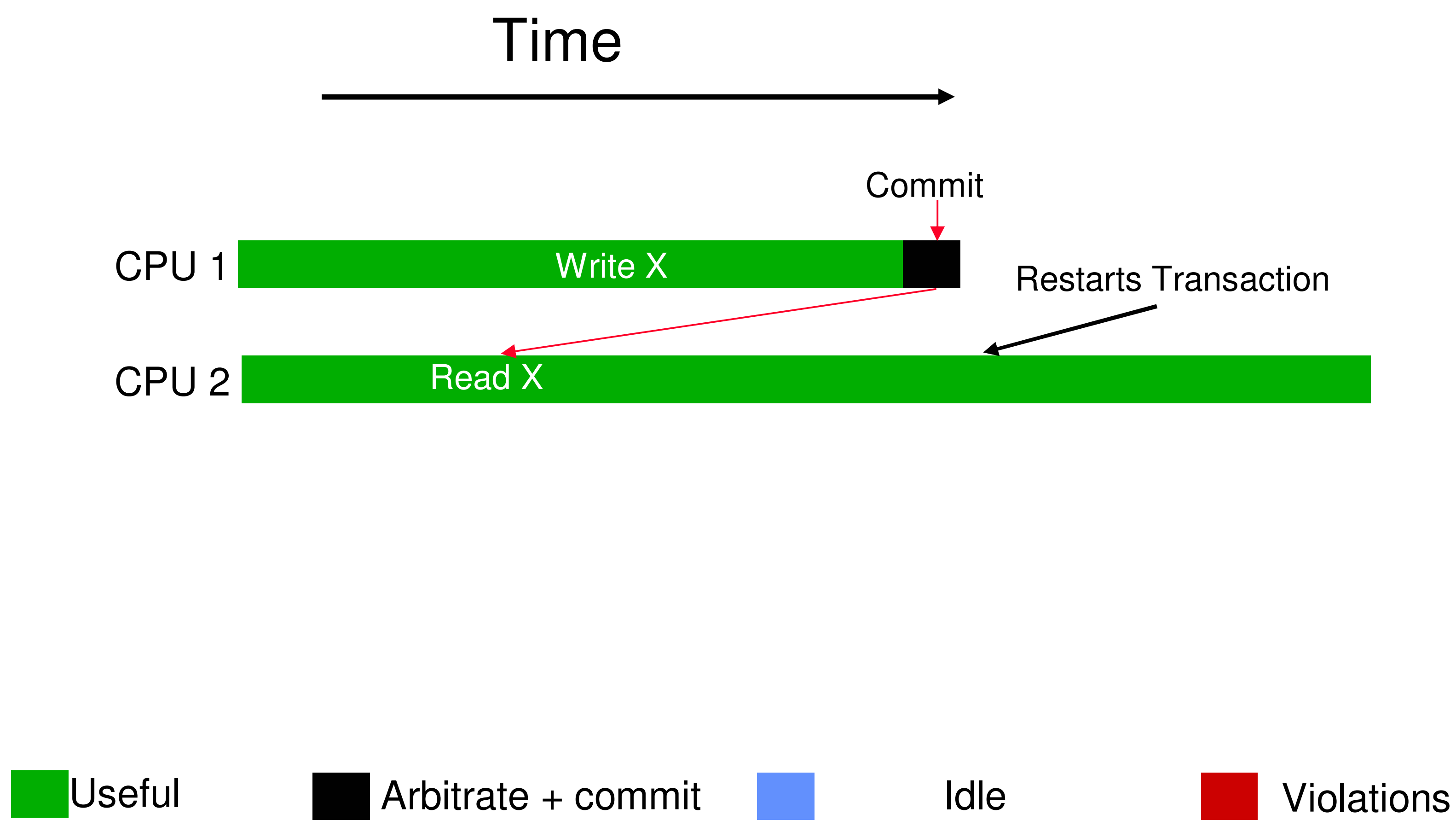


- Initial parallelization is quick and easy
- Performance tuning is critical

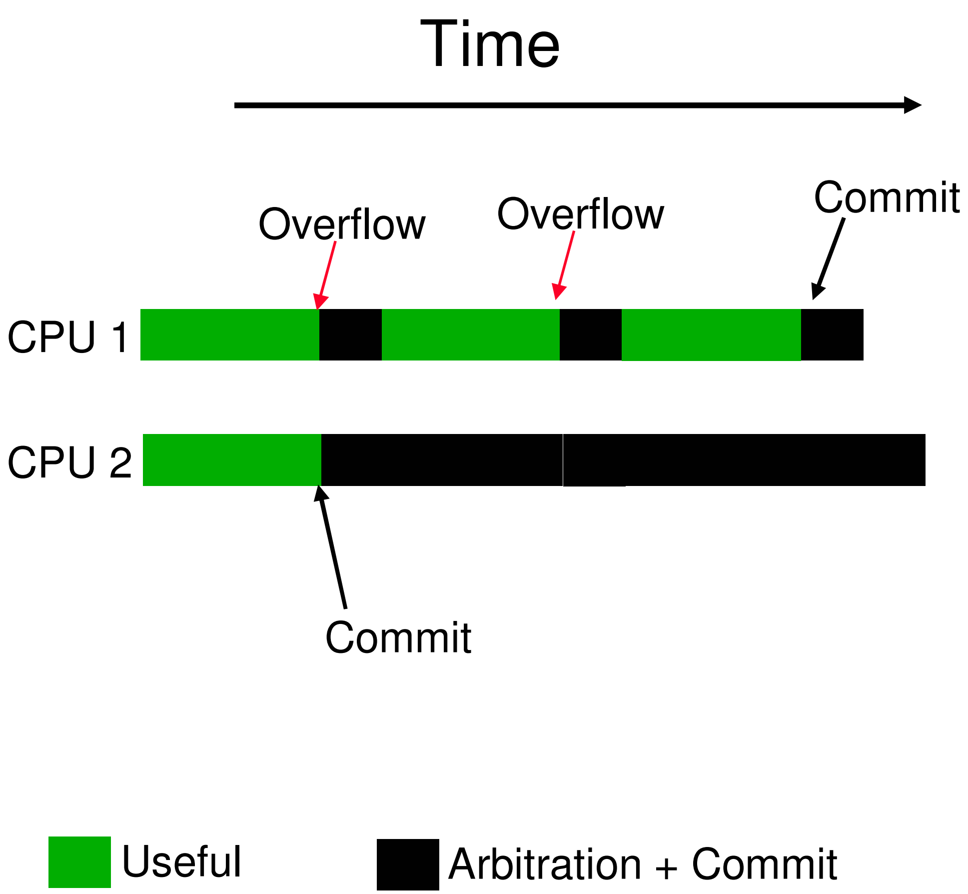
Performance Bottlenecks

- Dependency violations
  - Due to speculative nature of execution
- Buffer overflows
  - Transaction's state does not fit in cache
- Workload imbalance
  - Transactions are assigned disproportionate amount of work
- Transactional API overhead
  - Overhead of starting, committing, and aborting transactions

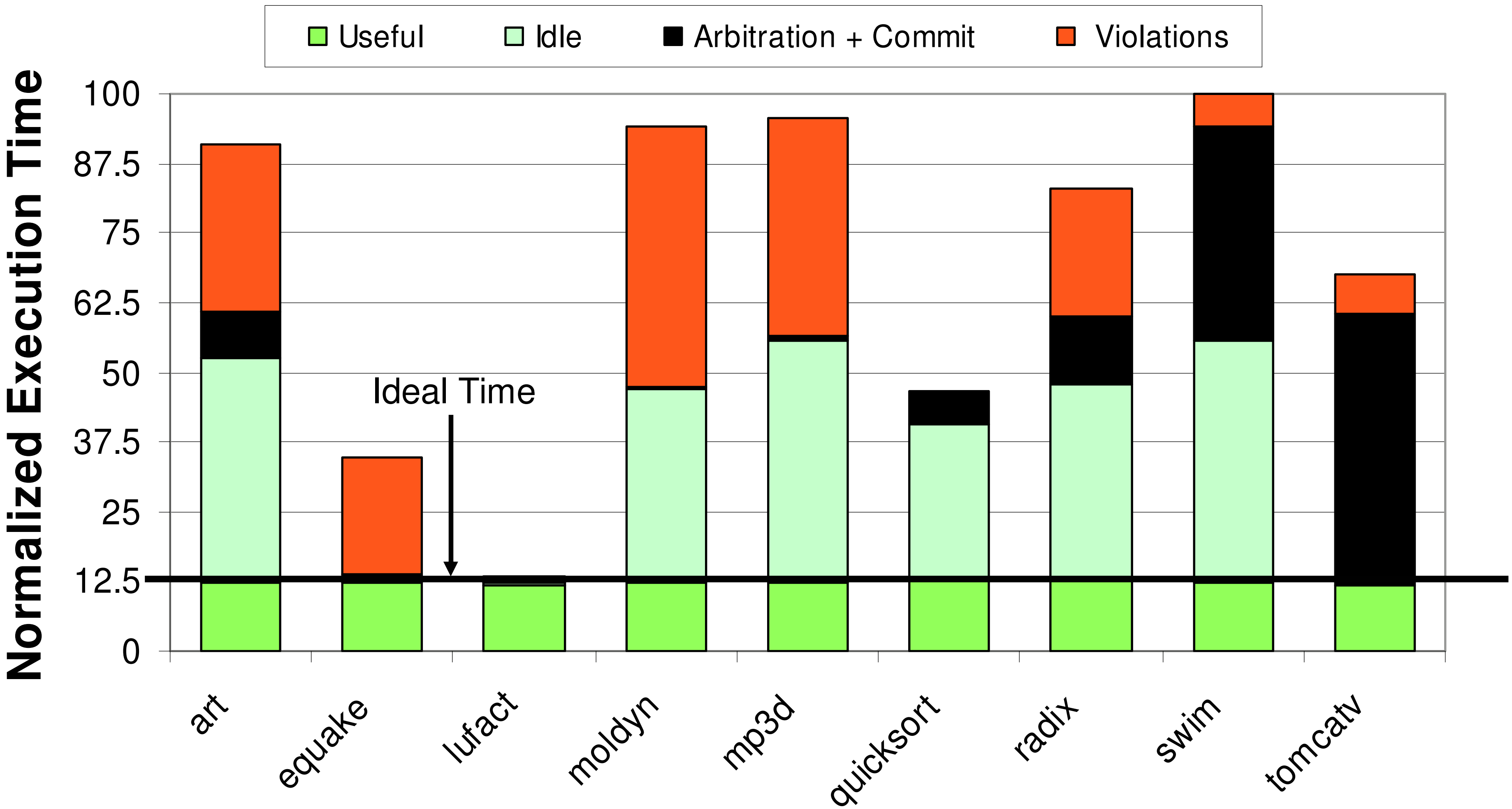
Dependency Violations



Buffer Overflows



Initial Performance Results - 8 processors



Outline

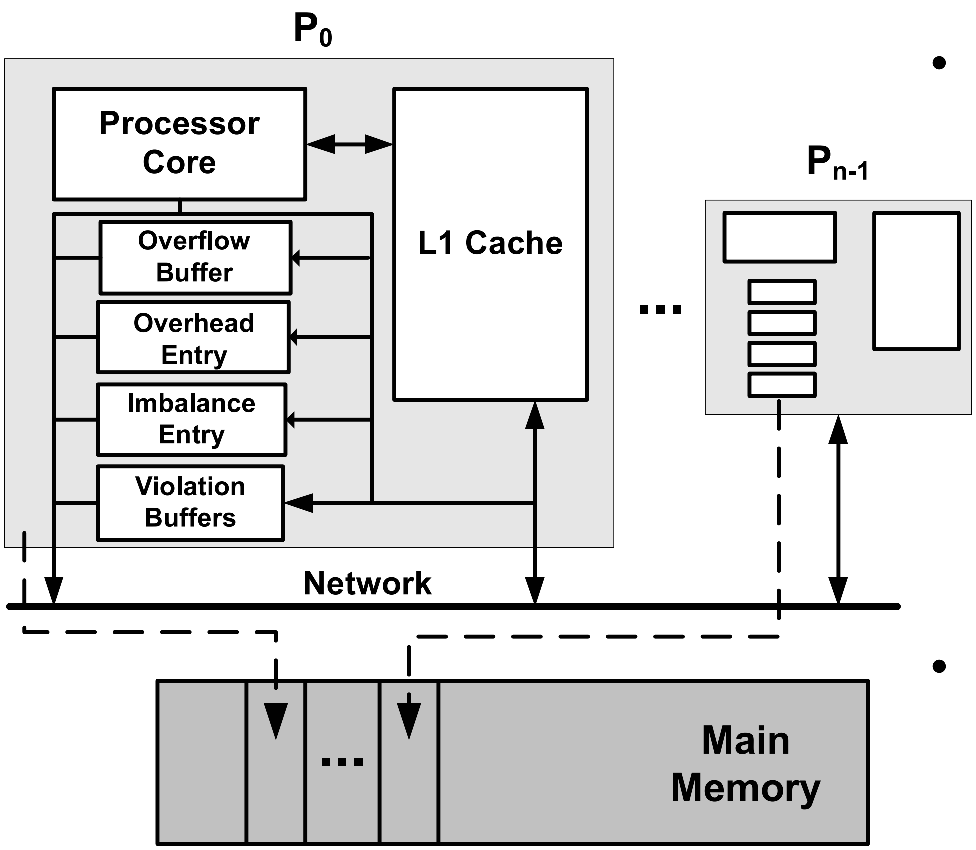
- Motivation
- TAPE system overview
- Example: Violation Profiling
  - Information gathering and filtering
  - Using profile information for optimizations
- Evaluation
- Conclusions

Key Insights

1. Leverage hardware for transactional execution
   - Already monitoring everything
   - TAPE operations can be amortized at commit time

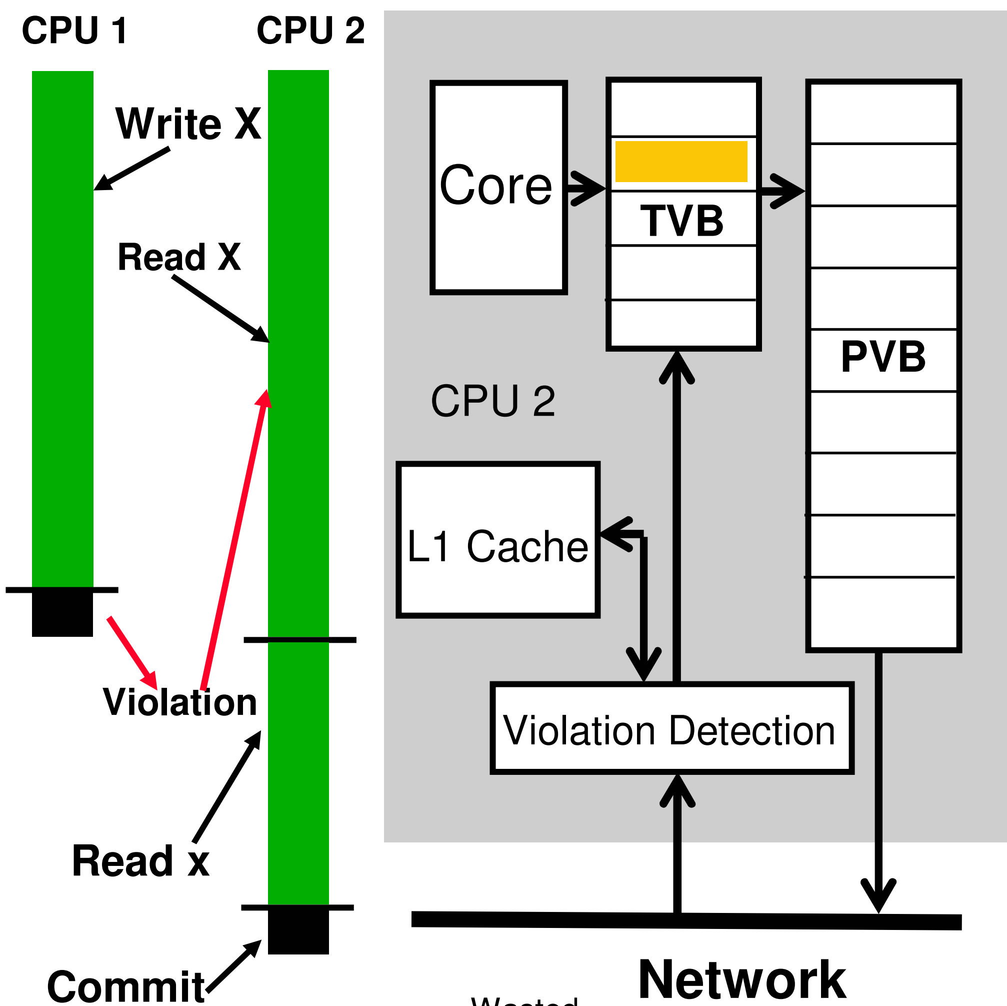
2. Repeatability of bottlenecks
   - Critical performance bottlenecks occur repeatedly
   - Data aggregation saves space without losing accuracy
   - TAPE automatically filters out infrequent bottlenecks

TAPE System Overview



- Online – Hardware
  - Each CPU gathers profile data in private buffers
  - Bottlenecks aggregated over multiple occurrences
  - Infrequent bottlenecks filtered out
  - Data periodically flushed to pre-allocated memory regions
- Offline – Software
  - Combine information from all CPUs
  - Rank bottleneck by cost
  - Format profiling output & relate data to source code

Profiling Violations



TPC	Wasted Work	Object addr	Read PC
PC_t	500	X	PC_2

- CPU-1 writes address X
- CPU-2 read address X
- CPU-1 commits first
- CPU-2 detects violation on X
  - Inserts entry in Transaction Violation Buffer
- CPU 2 restarts transaction
  - Re-reads address X
  - Sends read PC_2 to TVB
- CPU 2 commits
  - Most costly violations flushed to Period Violation buffer
  - Others may get evicted
- PVB can be flushed periodically

Example of Interaction with TAPE

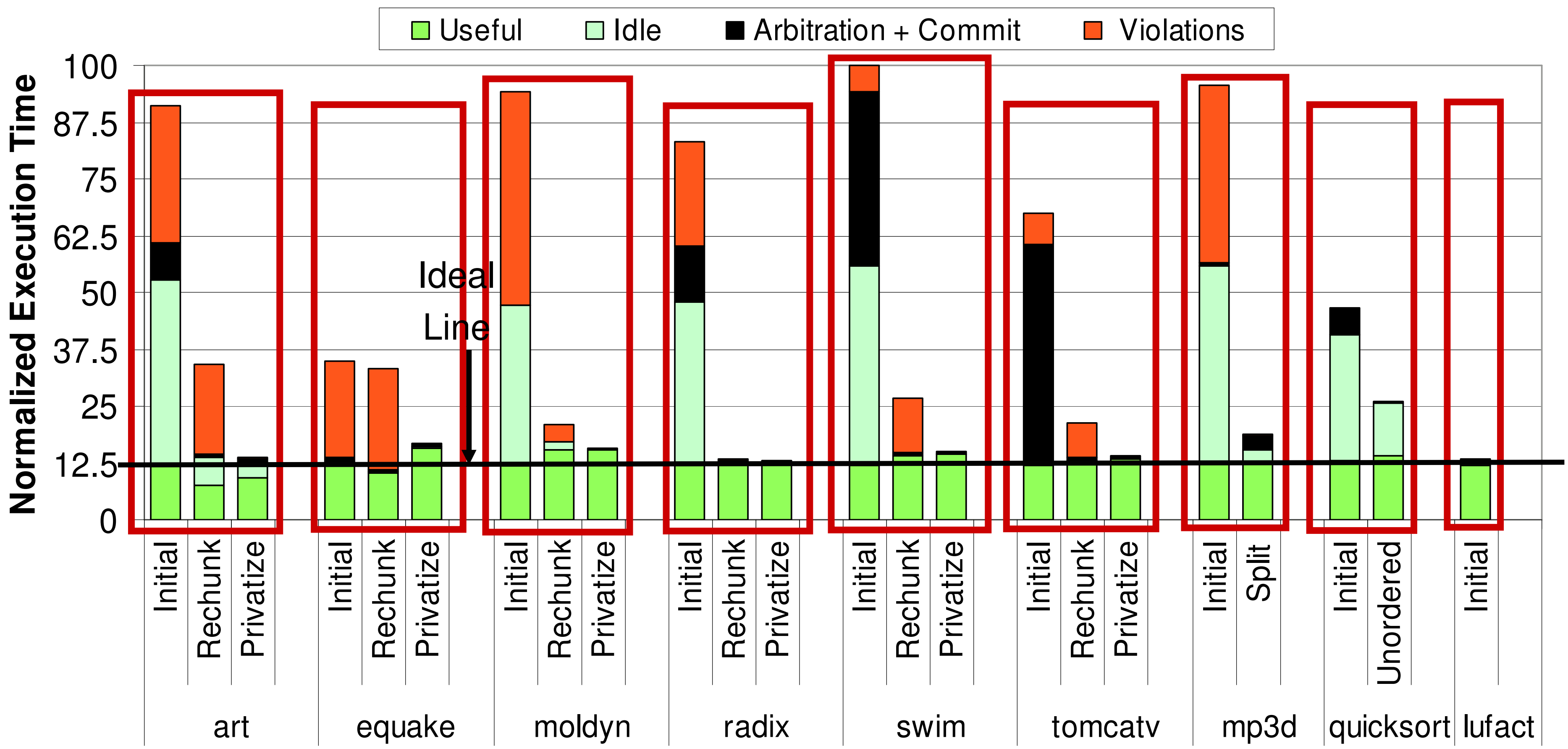
```
int* data = load_data();  /* input *
int i, buckets[101], sum = 0;

t_for_n (i = 0; i < 10000; i++; 500){
        pSum[TC_getMyID()] += data[i];
        buckets[data[i]]++;
}
for i = 0 to num_procs: sum += pSum[i];
print_buckets(buckets); /* output */
```

Violations

Evaluation Methodology

- 8-core CMP processor
  - Bus interconnected to shared L2 cache
  - Transactional buffering in private L1 caches (32 Kbytes)
  - Execution driven simulation with accurate contention modeling
- Applications: SPEC2K FP and SPLASH-2 benchmarks
  - See ASPLOS'04 for transactional programming details
- Questions
  - Ease of performance tuning with TAPE?
  - TAPE buffer size requirements
  - TAPE performance overhead

Performance Improvements for 8 Processors



- A maximum of two steps were required to fully optimize applications
- The programmer is directed to the source of the bottlenecks in the actual code

The Cost of TAPE

- Low Chip area cost
  - Proposed design point requires less than 5K SRAM bits, and 244 CAM bits per core
  - Less than 1% of overall chip area
- Low performance impact
  - Maximum slowdown of only 1.84% (Average was 0.28%)
  - Allows for continuous profiling, even on production runs
  - Maximum BW usage was 0.11%
- Memory Usage
  - On average only 1MB/hr of data generated

Conclusions

- TAPE: a profiling system for transactional applications
  - Support easy performance tuning
  - Complement correctness benefits of transactions
- Key features
  - Expressive: tracks all performance bottlenecks
  - Accurate: identifies bottleneck location in source code
  - Easy to use: leads to optimal performance in few tuning steps
  - Low overhead: negligible area & performance cost
  - Allows for continuous profiling, even on production runs

Thanks For listening

http://tcc.stanford.edu