Parrot: A Practical Runtime for Deterministic, Stable, and Reliable threads

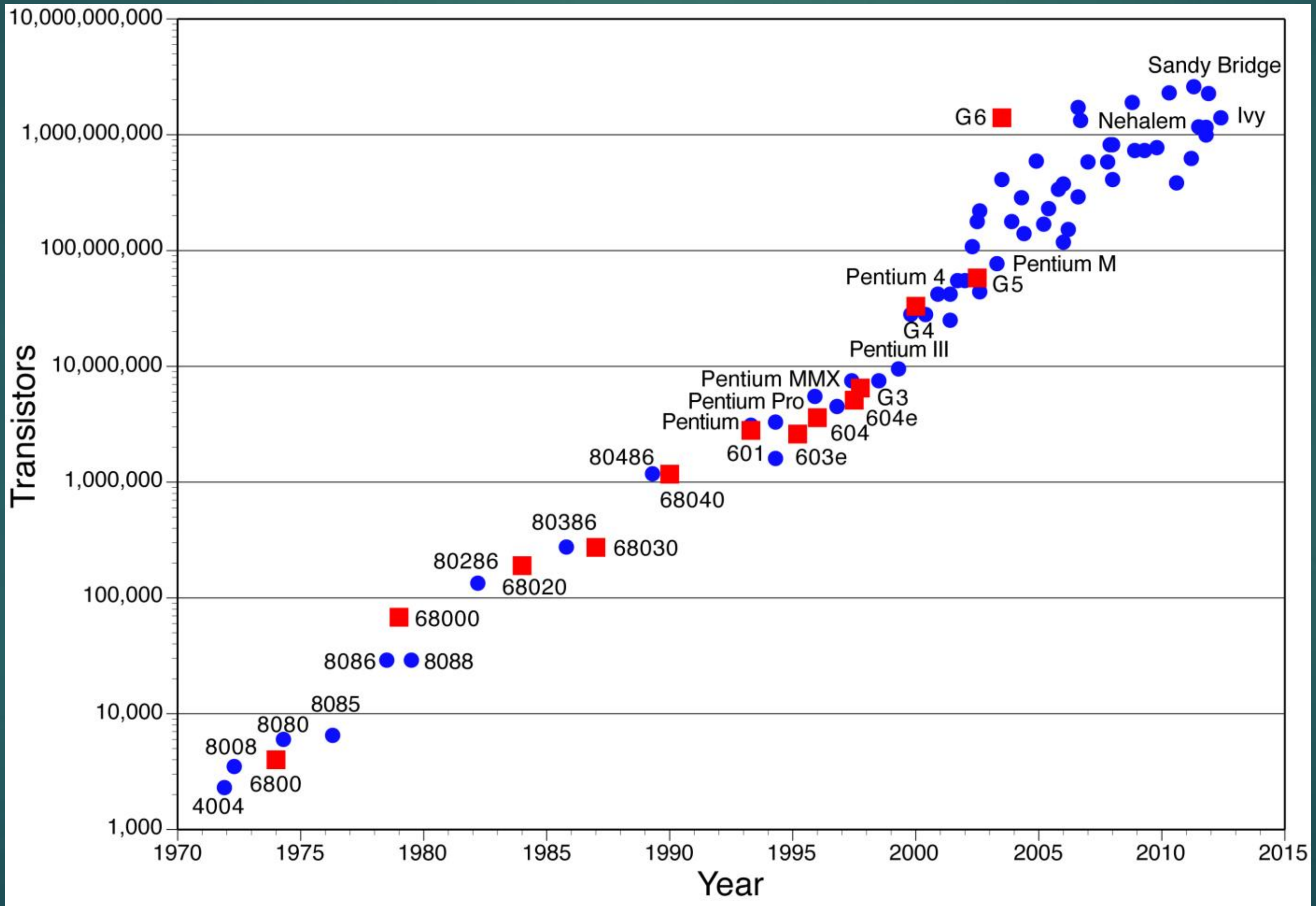
HEMING CUI, YI-HONG LIN, HAO LI, XINAN XU, JUNFENG YANG,
JIRI SIMSA, BEN BLUM, GARTH A. GIBSON, AND RANDAL E. BRYANT.

Presented by Ramachandra Pai

Outline

- Motivation
- Traditional and deterministic multithreading models
- What is stable multithreading models?
- PARROT: A Practical StableMT system
- How to use PARROT?
- Architecture
- Performance
- Evaluation

The one core era: Good times



Multiprocessor era

Parallelism gives improved performance but at cost of introducing complexity

- Deadlocks
- Race conditions
- Multiple threads accessing CS
- Non-determinism

Motivation

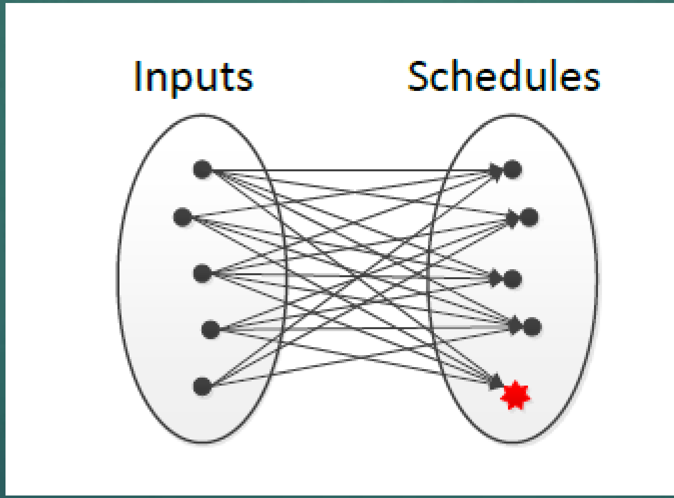
- Reliable parallelism is considered “something of a black art” because they are so hard to get right!

```
THREAD 1                    THREAD 2
lock(l);                    lock(l);
*p = . . . .;               p = NULL;
unlock(l);                  unlock(l);
```

```
// thread 1                 // thread 2
// deposit 100              // withdraw 100
t = balance + 100;
                            balance = balance - 100;
balance = t;
```

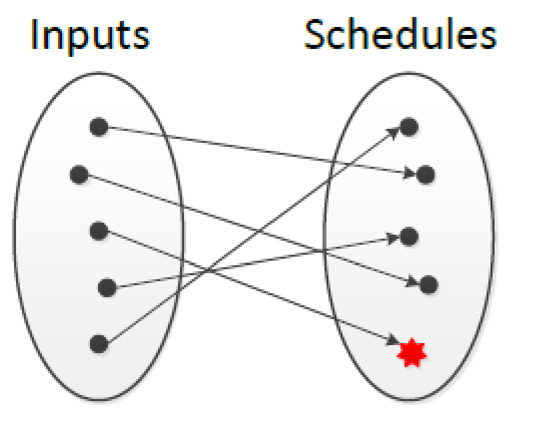
Traditional multithreading

- Many to Many mapping
- Hard to find concurrency bugs even if the buggy schedule is reproduced



How to reduce the order of threads?

- Deterministic Multithreading
  - Examples: Dthreads, Peregrine



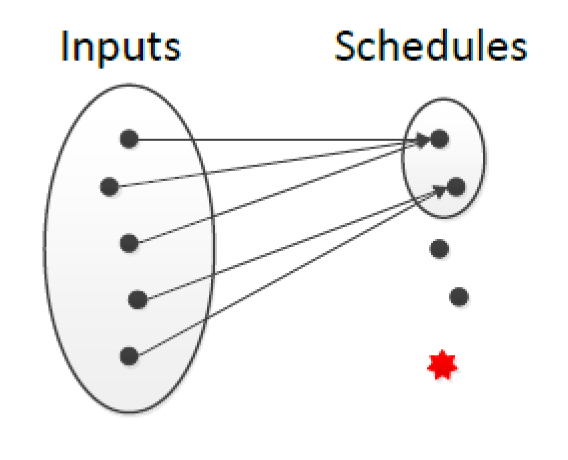
(b) *Deterministic.*

Is non-determinism the real culprit for all the problems?

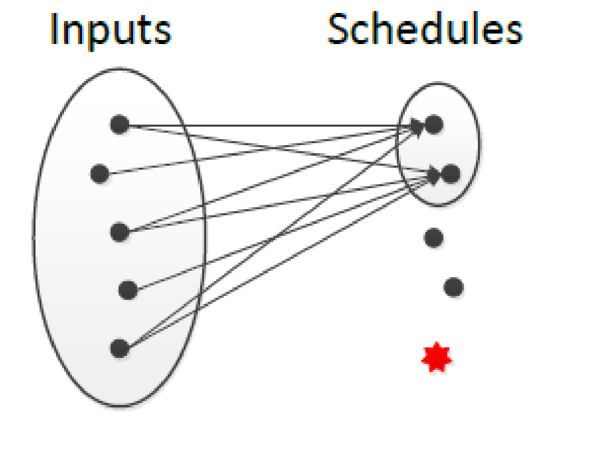
- Same input + same program -> same output.
- But what if the program changes slightly?
- We need stability for more reliable code. Hence we move to Stable multithreading models.

What is stable multithreading models?

- Reduces the number of schedules for all inputs
- Does so at the cost of performance.



(c) *Stable (deterministic).*



(d) *Stable (nondeterministic).*

PARROT: A StableMT model

- Reduction in schedules
  - Round robin scheduling.
- How do we get performance?
  - **Soft barriers**: “parallel scheduling of chosen computations”
  - **Performance critical sections**: “Ignore determinism”
- Integrated with **DBUG**

Performance hints

- Soft Barriers:
  - Encourages scheduler to co-schedule a group of threads
  - Scheduler may ignore it if it affects correctness

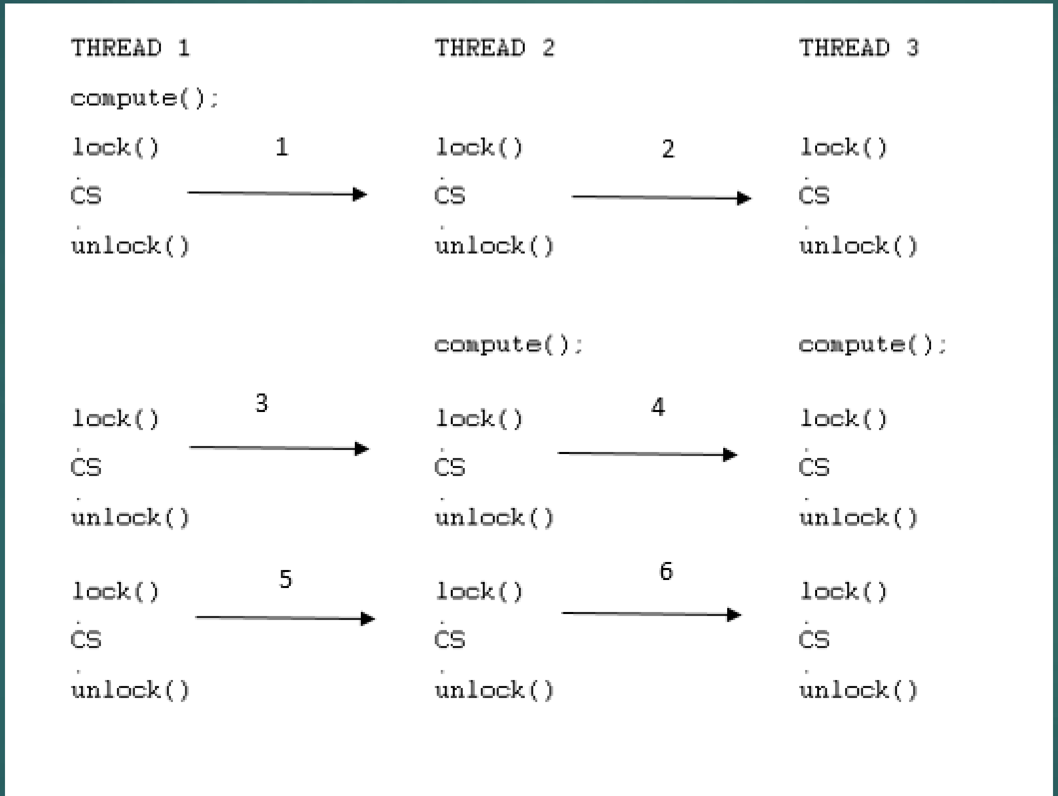
  void soba_init(int groupsize, void *key, int timeout);

  void soba_wait(void *key);
- Performance Critical section:
  - Removes the round robin scheduling
  - Allows OS to schedule this part of code.
  - Introduces non-determinism.

Example:

THREAD 1	THREAD 2	THREAD 3
compute();		
lock()	lock()	lock()
.	.	.
CS	CS	CS
.	.	.
unlock()	unlock()	unlock()
	compute();	compute();
lock()	lock()	lock()
.	.	.
CS	CS	CS
.	.	.
unlock()	unlock()	unlock()
lock()	lock()	lock()
.	.	.
CS	CS	CS
.	.	.
unlock()	unlock()	unlock()

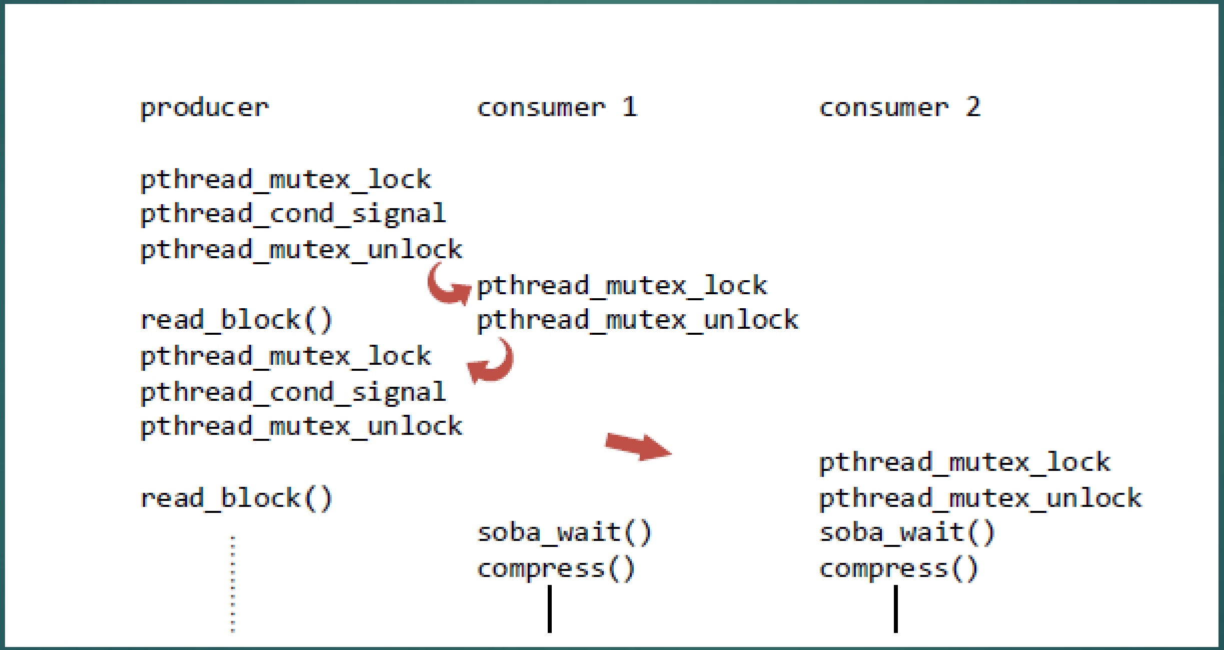
Example: total order of events



How to use PARROT?


```
int main(int argc, char *argv[]) {
  ...
  soba_init(nthreads); /* performance hint */
  for (i = 0; i < nthreads; ++i)
    pthread_create(..., NULL, consumer, NULL);
  for (i = 0; i < nblocks; ++i) {
    char *block = read_block(i);
    pthread_mutex_lock(&mu);
    enqueue(q, block);
    pthread_cond_signal(&cv);
    pthread_mutex_unlock(&mu);
  }
  ...
}
void *consumer(void *arg) {
  while(1) {
    pthread_mutex_lock(&mu);
    while (empty(q)) // termination logic elided for clarity
      pthread_cond_wait(&cv, &mu);
    char *block = dequeue(q);
    pthread_mutex_unlock(&mu);
    ...
    soba_wait(); /* performance hint */
    compress(block);
  }
}
```

Total order of events

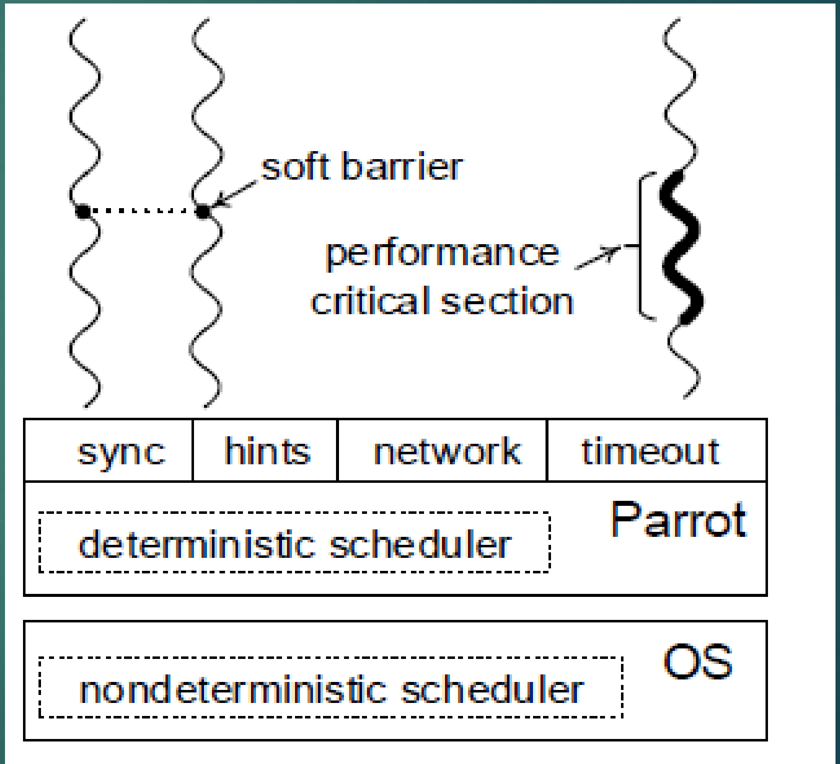


What is DBUG?

- Model checking model : checks all the states of a system
- Mutually beneficial to both systems
  - Parrot Reduces the number of schedules. Hence reducing the checking sample space.
  - DBUG helps check schedules that matter to Parrot and developers.

Architecture:

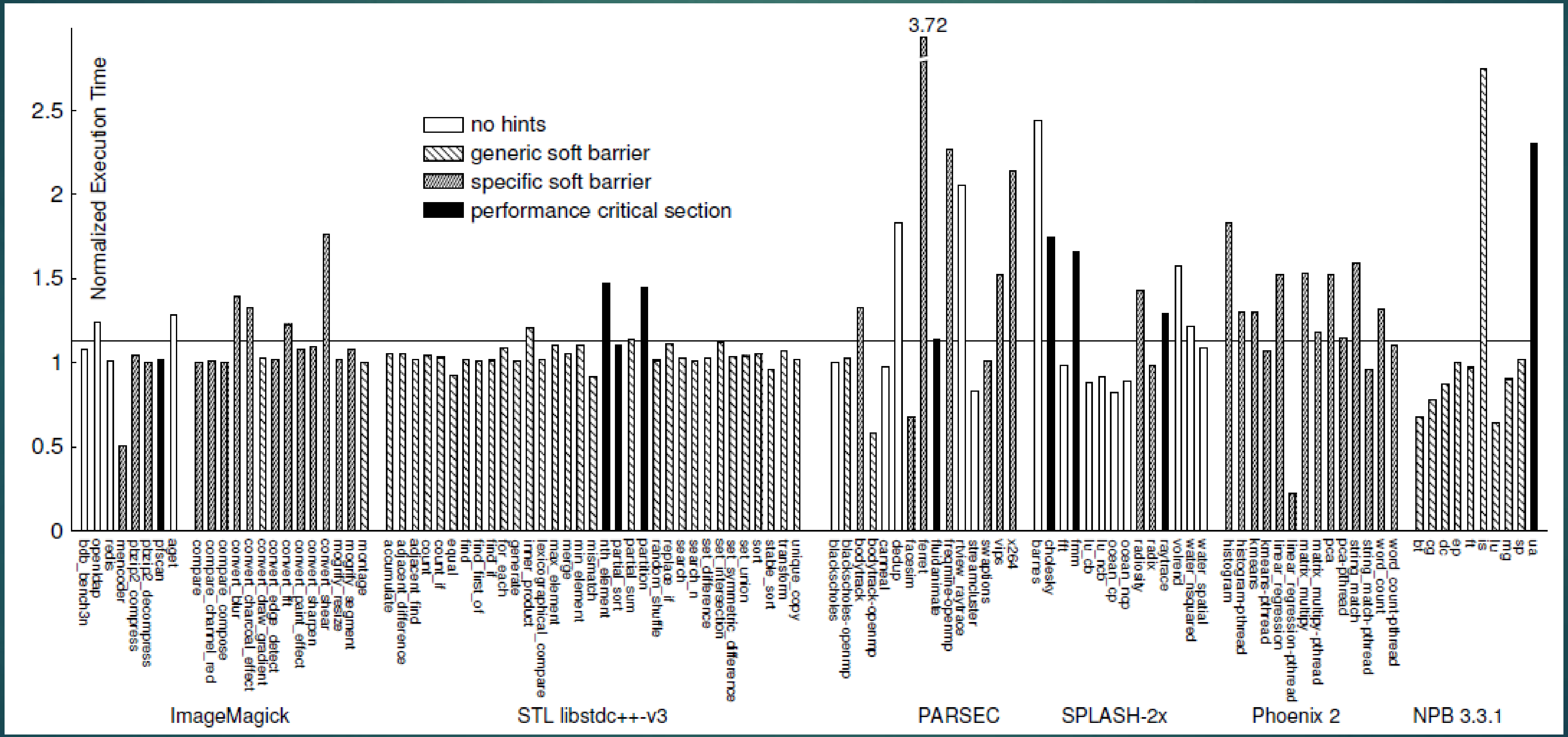
- Deterministic Scheduler
- Performance hints
- Wrapper functions for pthread
- Network
- Timeout



How does parrot perform in the real world?

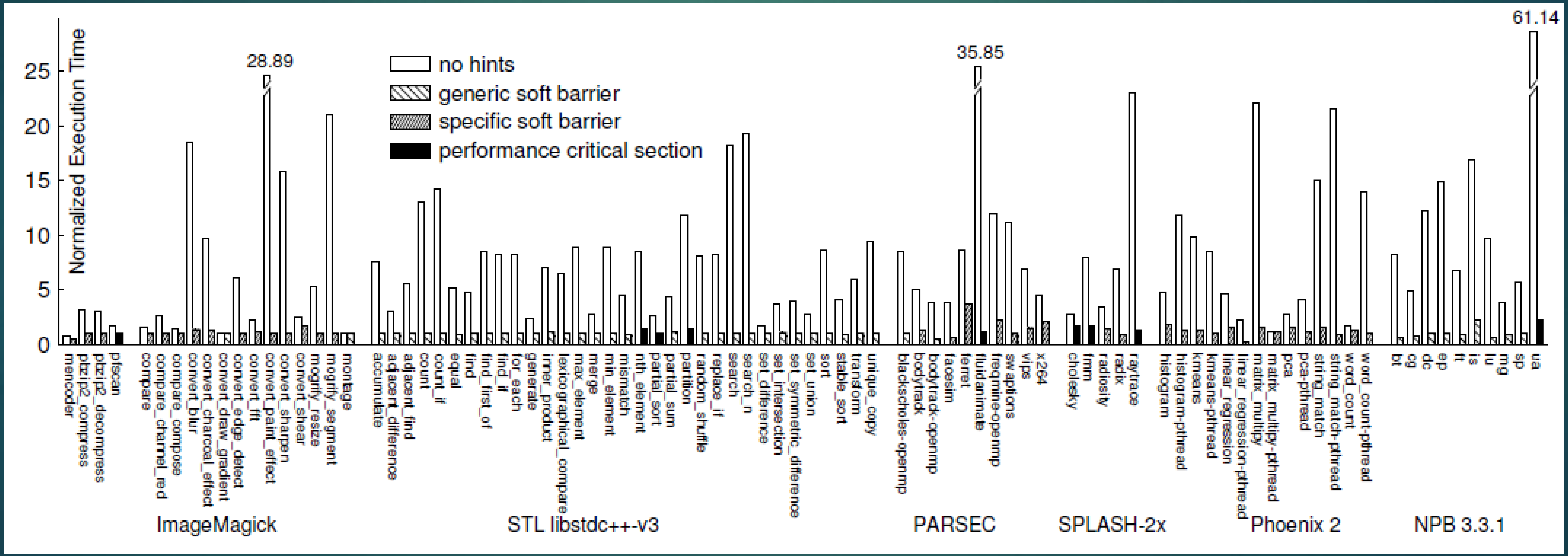
- 55 Real world programs
  - BerkleyDB, database Library
  - OpenLDAP, server with Lightweight directory Access protocol
  - Mplayer, video encoder/decoder and player
  - Pbzip2, a parallel compression utility etc.
- 53 programs used in benchmarks
  - 15 program in PARSEC
  - 14 in phoenix etc.

Performance charts



Effects of Soft barriers and Performance critical sections

- Reduction of overhead from 510% to 11.9%



Evaluation

- Easy to use
- Performance takes a hit, and sometimes its too bad.
- Better than its predecessors in terms of stability and performance. e.g.: Dthreads, Peregrine
- Deterministic
- Does not solve data races
- Easily deployable
- Replay Debugging

Thank you. Any Questions?