CPU自旋锁优化

2019年10月14日

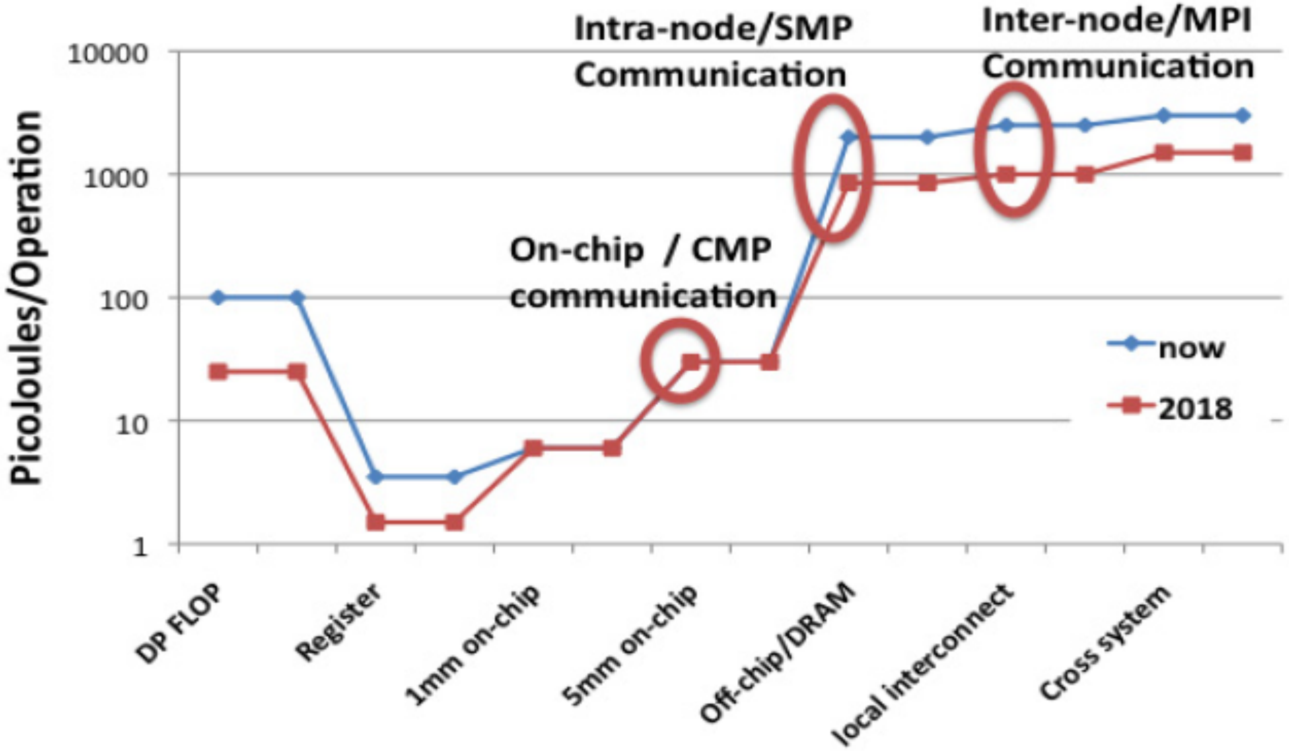
蚂蚁金服
ANT FINANCIAL

内容

- 数据迁移问题 & 对性能影响
- Spinlock Issues on Multi-Socket systems
  - MCS Spinlock
  - Critical Section Integration
  - NUMA Aware Spinlock
- NUMA Spinlock
- NUMA Spinlock Example
- NUMA Spinlock Result

数据迁移问题&对性能影响

数据迁移是体系架构的主要瓶颈



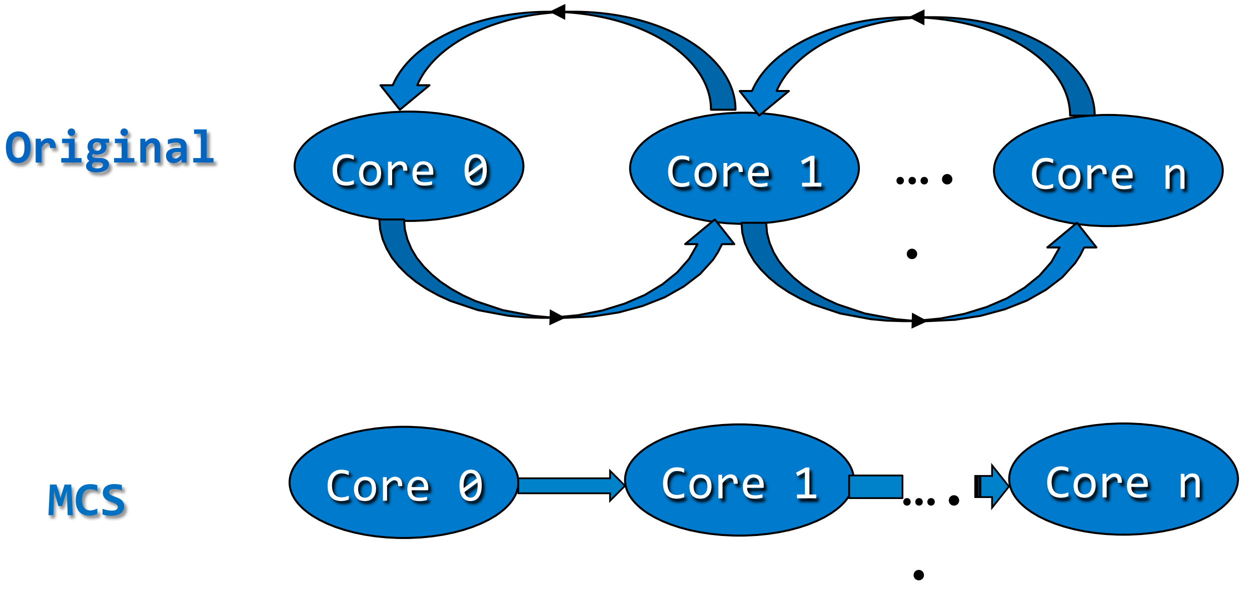
双浮运算需要 10PJ
数据移动需要 1000PJ

优化自旋锁

- On multi-socket multiple core systems,
  Access Latency:

  remote socket > local socket > sibling core > local core

MCS Spinlock



MCS helps us to reduce useless lock movement in spinning stage.

Critical Section Integration(CSI)

```
LOCK X
result =
CS(A);
UNLOCK X
print result
```

Core 0

```
void * work(void *V)
{
  v->result = CS(v->A);}
  Acquire LOCK
  If(contention) {
    send work
    while(result);
  }else {
    CS(A);
    If(incoming request)
      run work list
    Release LOCK
  }
  print result
```

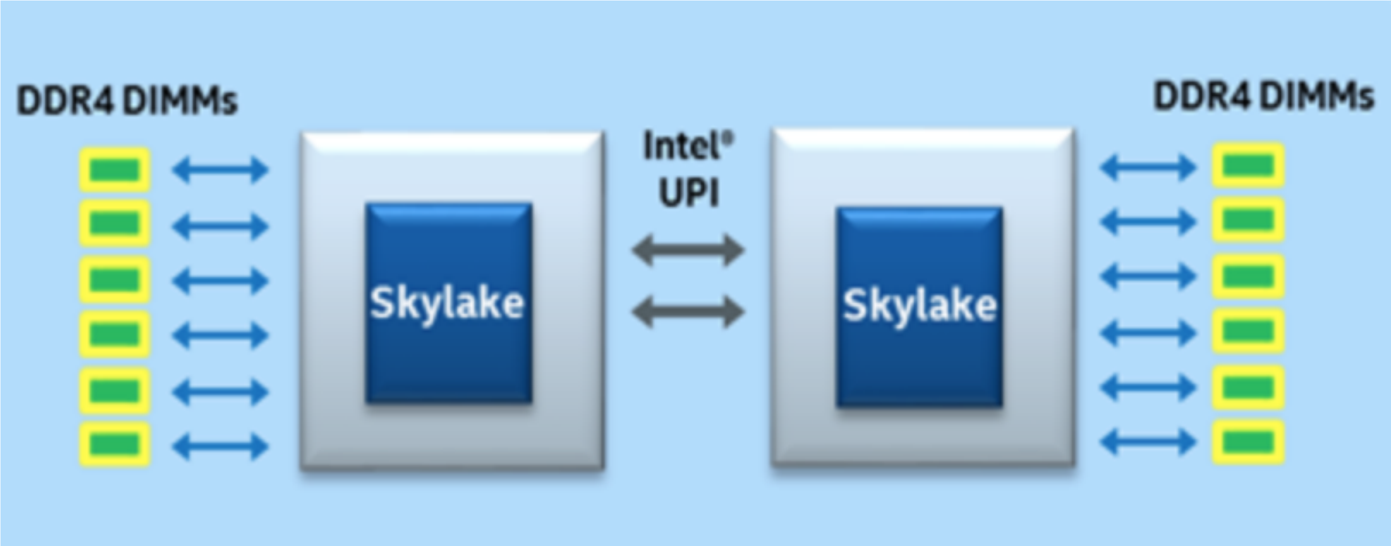
Core n

```
void * work(void *V)
{
  v->result = CS(v->A);}
  Acquire LOCK
  If(contention) {
    send work
    while(result);
  }else {
    CS(A);
    If(incoming request)
      run work list
    Release LOCK
  }
  print result
```

CSI helps us to reduce share data movement.

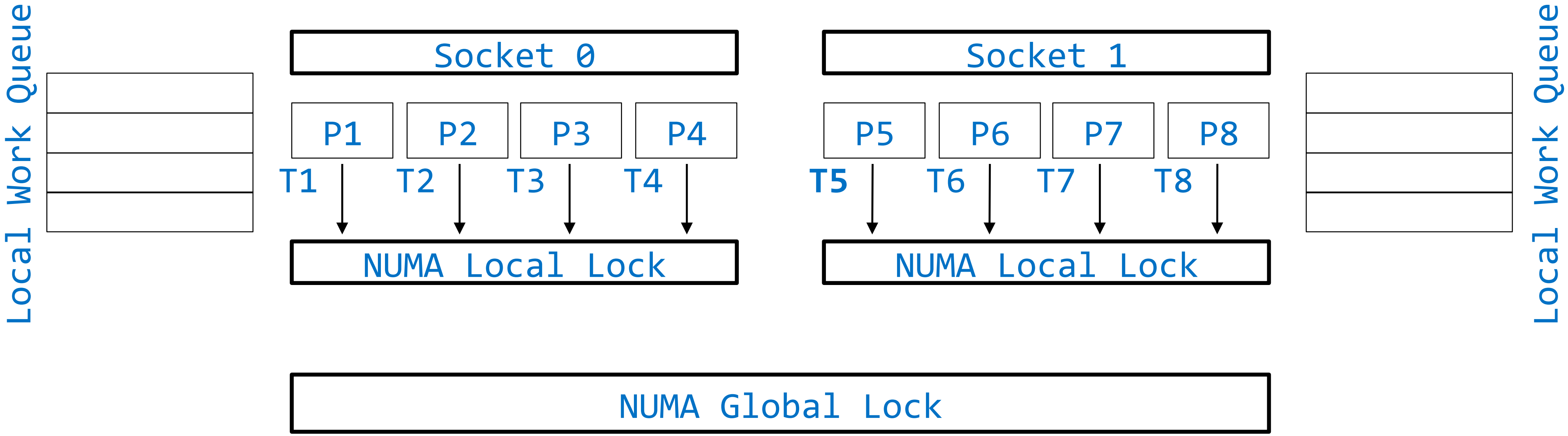
NUMA Aware Spinlock(NAS)

```
Acquire LOCK from Socket 0/1
  if (contention)
    append work list & waiting

Acquire Global LOCK
  if (contention)
    waiting

Run local work list

Release LOCK from Socket 0/1

Release Global LOCK
```

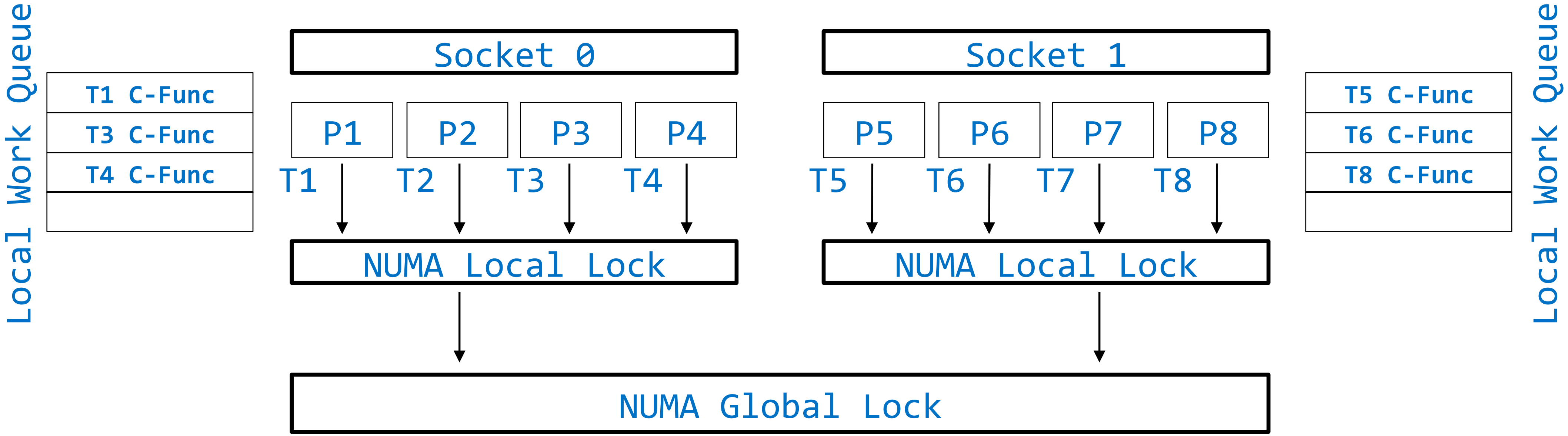


NAS helps us to reduce off-chip lock movement.

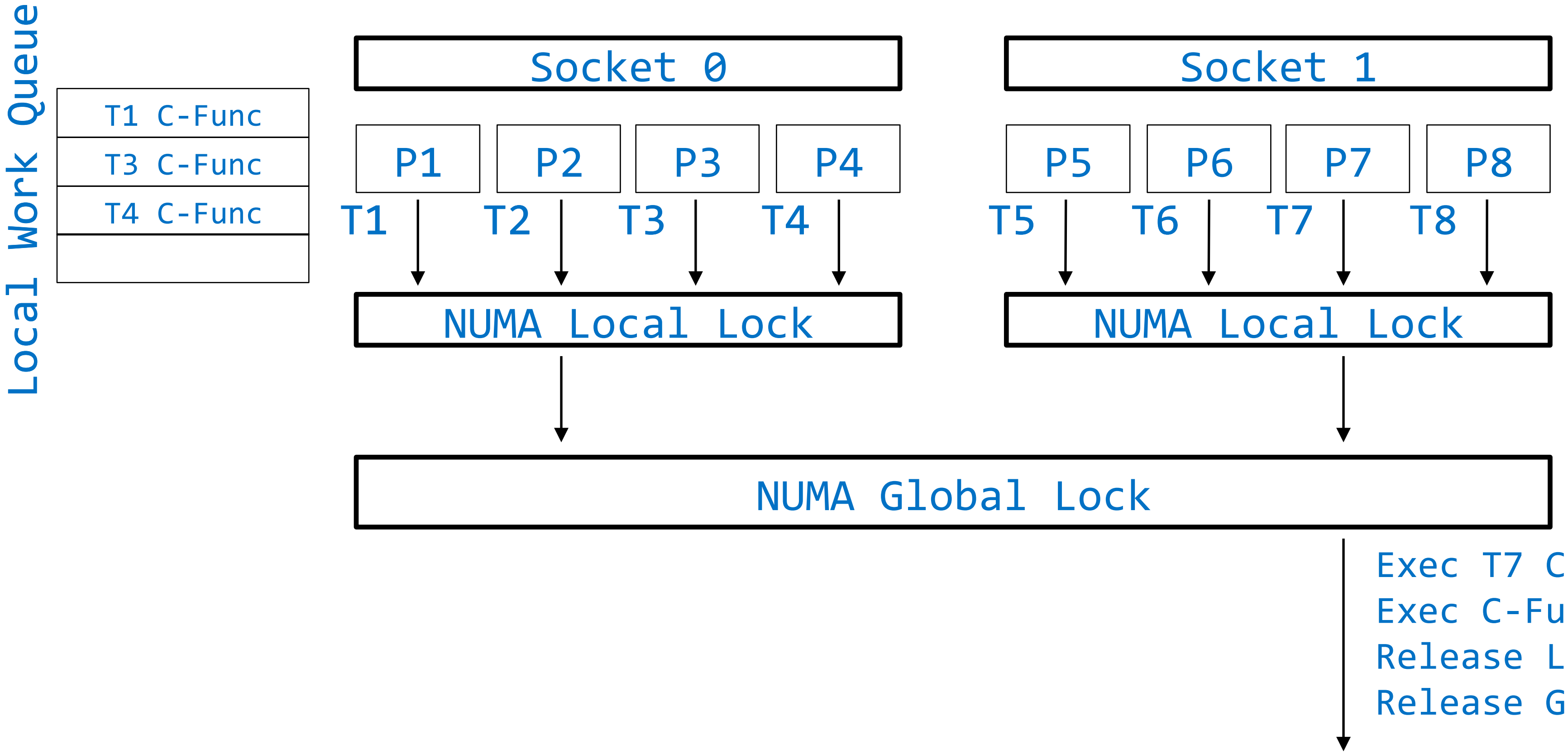
NUMA Spinlock



NUMA Spinlock (2)

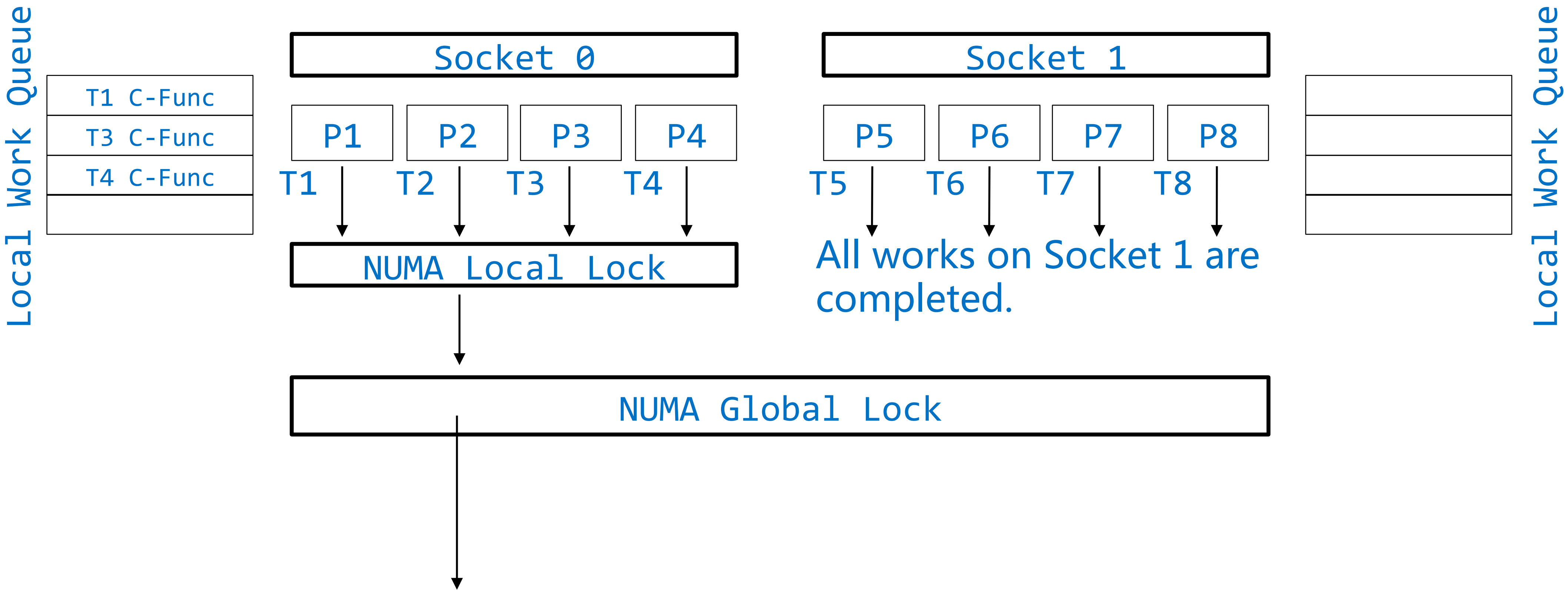


NUMA Spinlock (3)



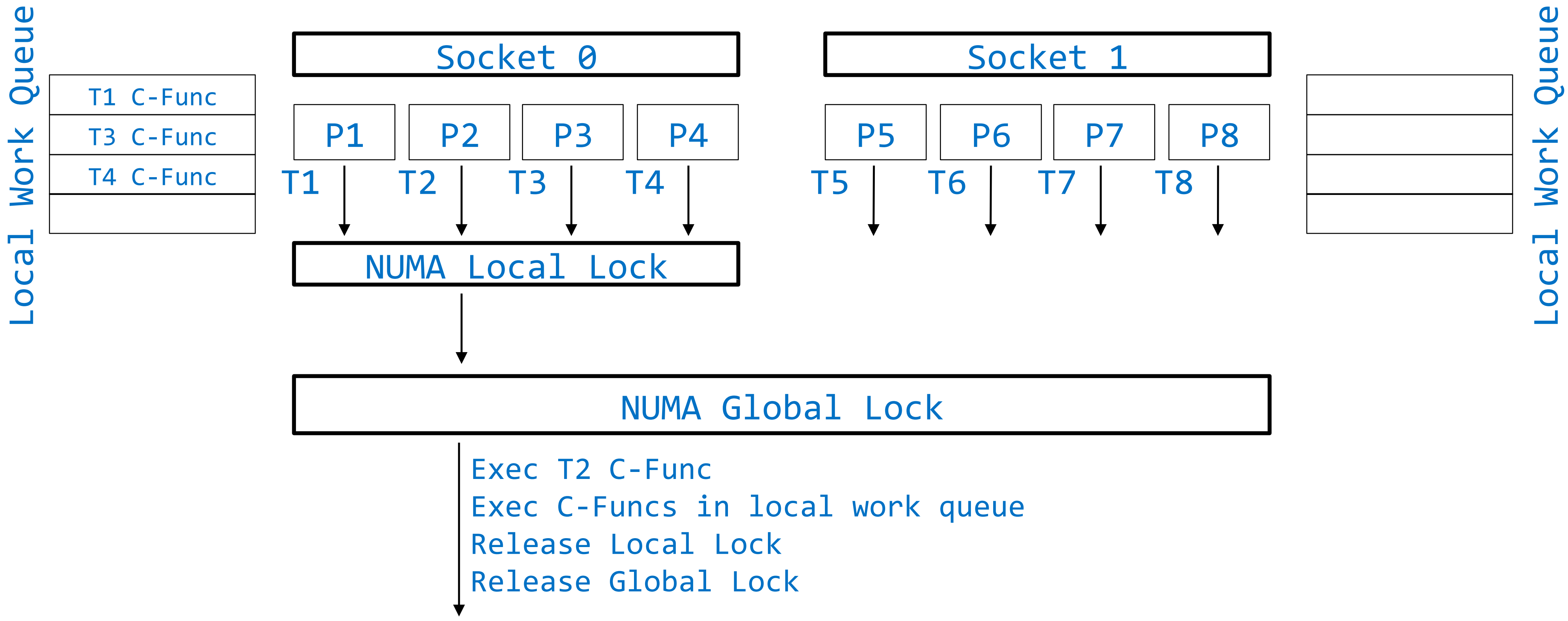
Exec T7 C-Func
Exec C-Funcs in local work queue
Release Local Lock
Release Global Lock

NUMA Spinlock (4)



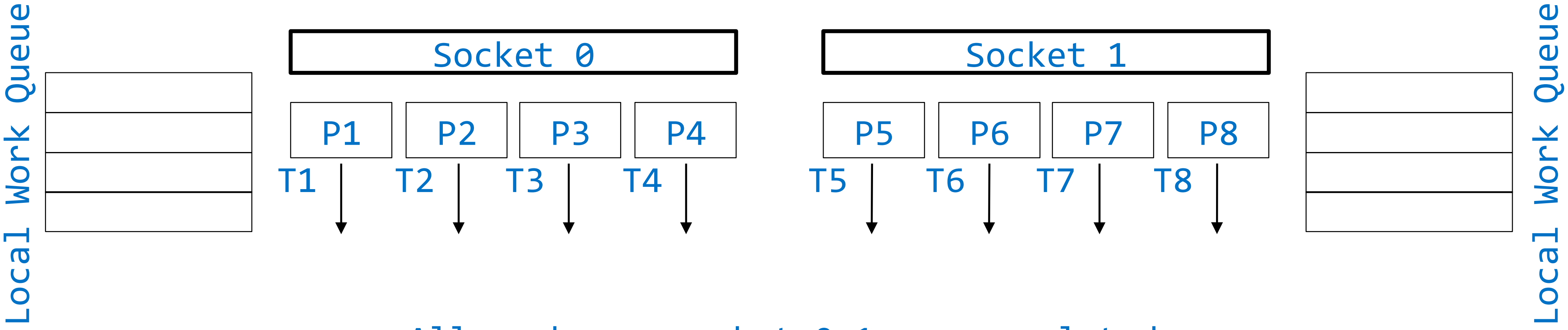
All works on Socket 1 are completed.

NUMA Spinlock (5)



Exec T2 C-Func
Exec C-Funcs in local work queue
Release Local Lock
Release Global Lock

NUMA Spinlock (6)



All works on socket 0,1 are completed.

NUMA spinlock helps us to minimize cross-socket traffic as well as localize the serialized workload to one core for execution.

NUMA Spinlock Example

```
struct numa_spinlock *lock;

work_thread (void *arg) {
  struct work_todo_argument
work_todo_arg;
  struct numa_spinlock_info lock_info;
  if (numa_spinlock_init (lock,
&lock_info)) {
      printf ("numa_spinlock_init
failure: %m\n");
      exit (1);
  }
  work_todo_arg.arg = arg;
  lock_info.argument = &work_todo_arg;
  lock_info.workload = work_todo;
  numa_spinlock_apply (&lock_info);
  return lock_info.resul
}
```

A ‘numa_spinlock’ pointer uniquely identifies a NUMA spinlock object.

This data type uniquely identifies a NUMA spinlock information object for a thread.

Initialize the NUMA spinlock information block pointed to by INFO with a NUMA spinlock pointer LOCK. The return value is ‘0’ on success and ‘-1’ on failure.

A pointer to argument passed to the WORKLOAD function pointer.

A function pointer to the workload function serialized by spinlock.

Apply for spinlock with a NUMA spinlock information block pointed to by INFO.

When ‘numa_spinlock_apply’ returns, the spinlock is released and the RESULT member of INFO contains the return value of the WORKLOAD member.

NUMA Spinlock Example (2)

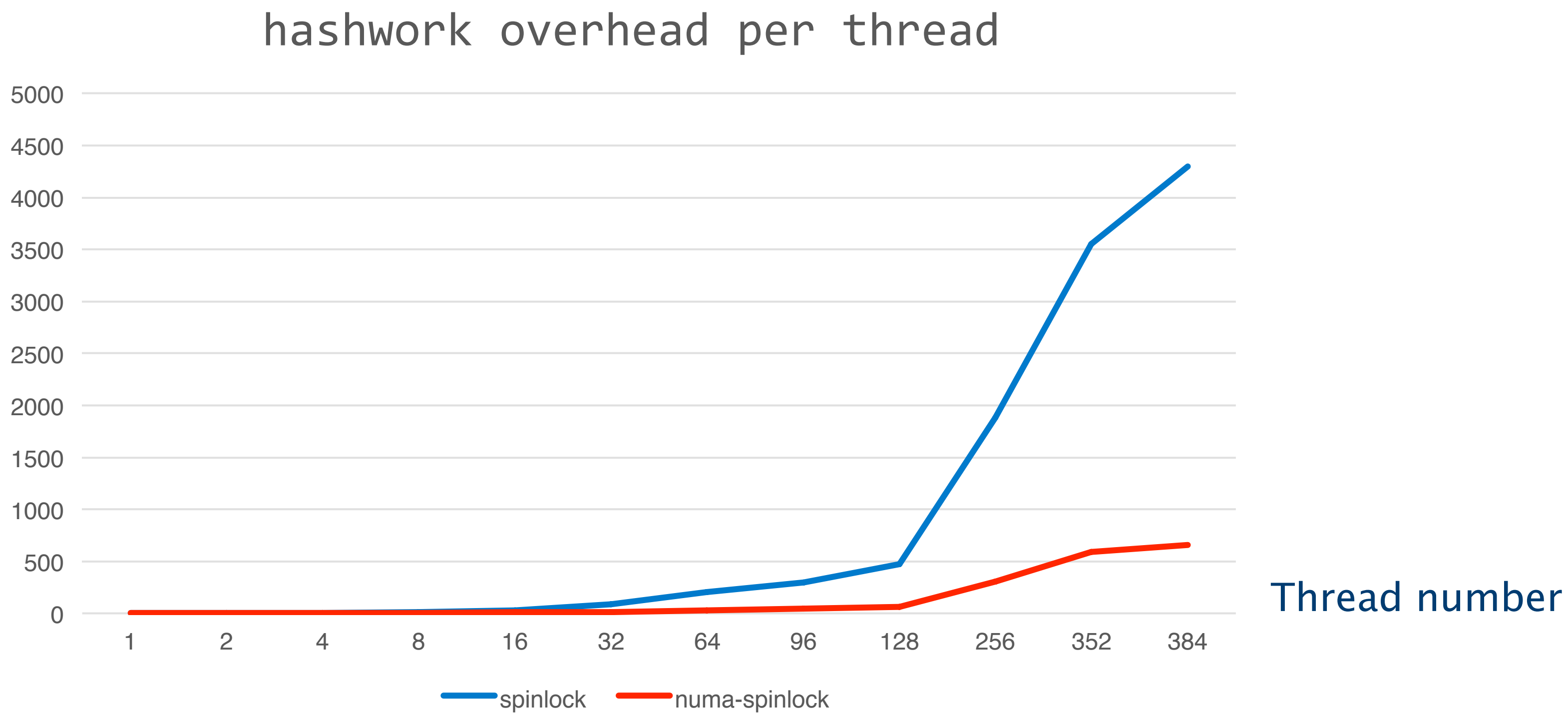
```
• static void *work_todo (void *v) {
•   struct work_todo_argument *p = v;
•   void *ret_val;
•   ret_val = serialized_work (p->arg);
•   return ret_val;
• }
```

Do the real work with p->arg.

Return value is set to lock_info.result.

```
static void __attribute__((constructor)) init_numaspinlock (void) {
  lock = numa_spinlock_alloc ();
}

static void __attribute__((destructor)) destroy_numaspinlock (void) {
  numa_spinlock_free (lock);
}
```

Micro-Benchmark Result



已经提交硬件优化，解决最后的问题，基本达到理论值，自旋锁问题即将解决

1%的串行化导致 64核心只有 40核心的结果 Speed up = T / ((0.99T/64) + 0.01T) = 39.2倍

因此建议尽量不要使用同步机制，不用性能才最好 😄

THANK YOU!

蚂蚁金服
ANT FINANCIAL