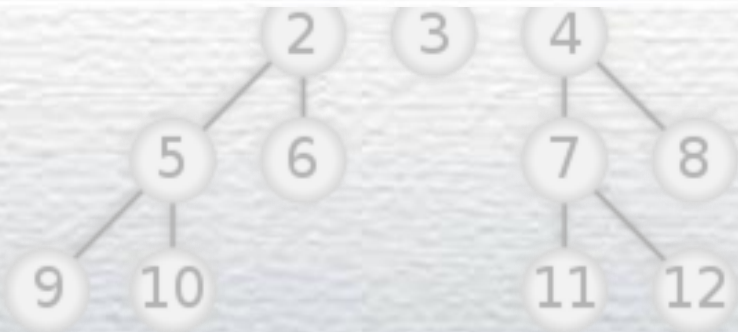
图广度搜索算法并行化方法研究

Study on Parallelization of Breadth-first Search Algorithm

计算机　　程亦超

指导教师　孙海平



图广度优先搜索算法的并行化方法研究

我做了哪些工作？

平台	体系结构	编程模型	优化技术	数据（图）	算法
CPU	Intel Nehelam	OpenMP	1. 位图 2. 两级队列 3. 预读 4. Socket队列	1. 规则图 2. 不规则图 3. 真实世界网络 4. 小世界网络	$O(V \cdot L + E)$ $O(V + E)$
GPU	NVIDIA Fermi	NVIDIA CUDA	1. 两级队列 2. 预读		$O(V + E)$

图广度优先搜索算法的并行化方法研究

做这些有什么意义？

- 图算法的广泛应用
- 多核时代已经到来
- 图算法难以并行化

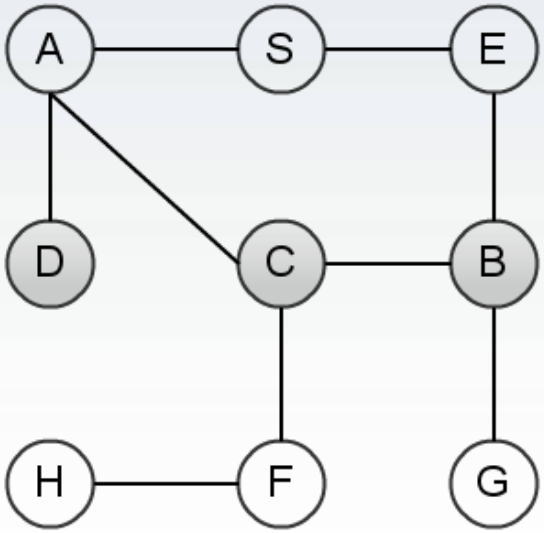
图广度优先搜索算法的并行化方法研究

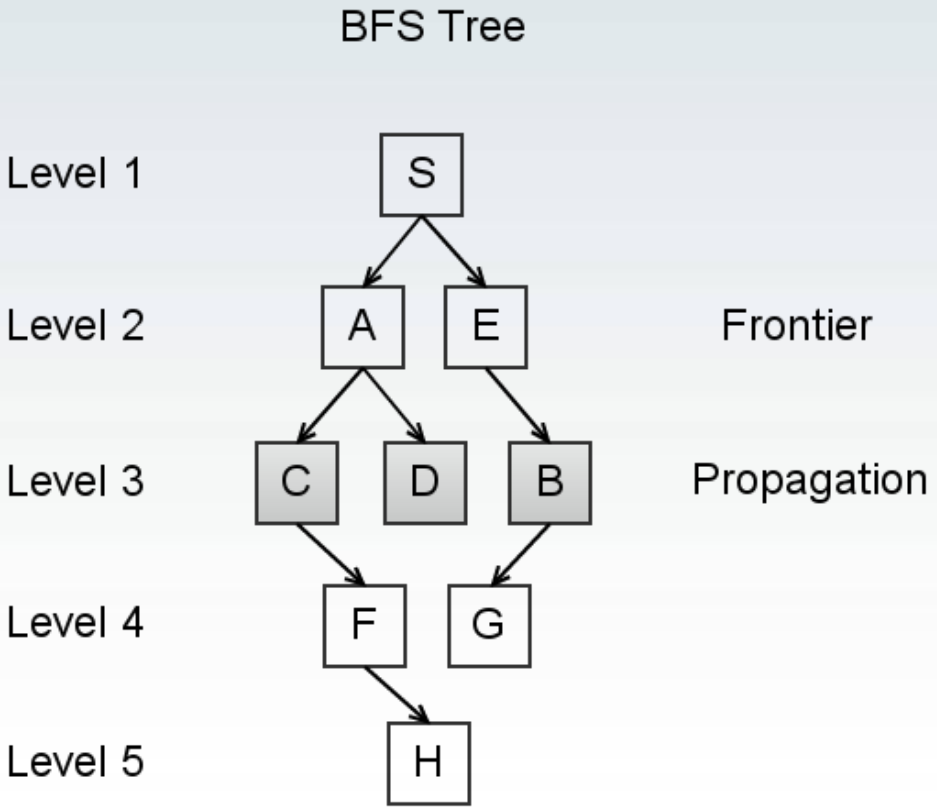
为什么要用这种方法做？

图广度优先搜索算法的并行化方法研究

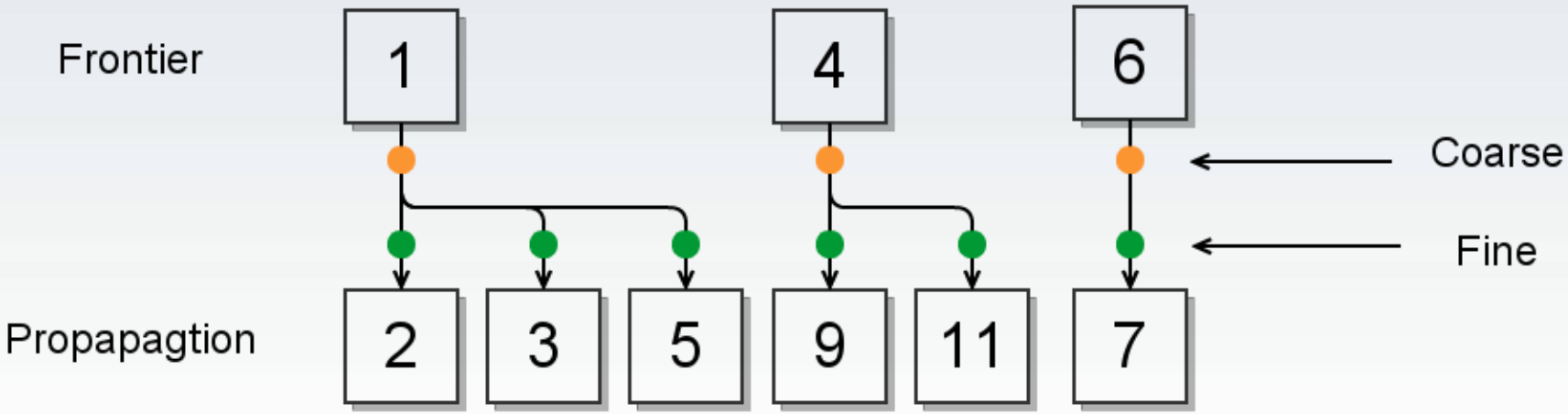
介绍 | BFS | 优化 | 实验 | 总结

串行BFS算法





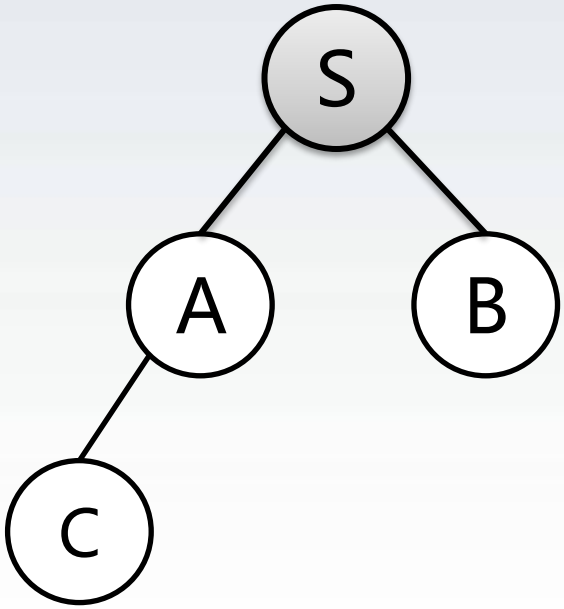
BFS算法的并行性

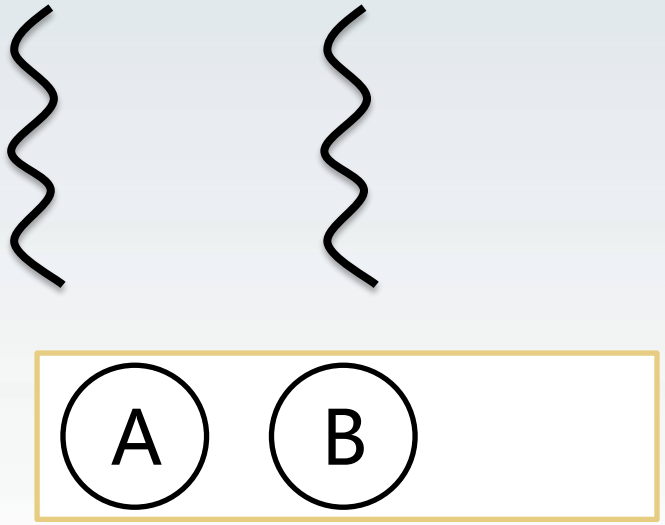


两种颗粒度的并行性

图广度优先搜索算法的并行化方法研究

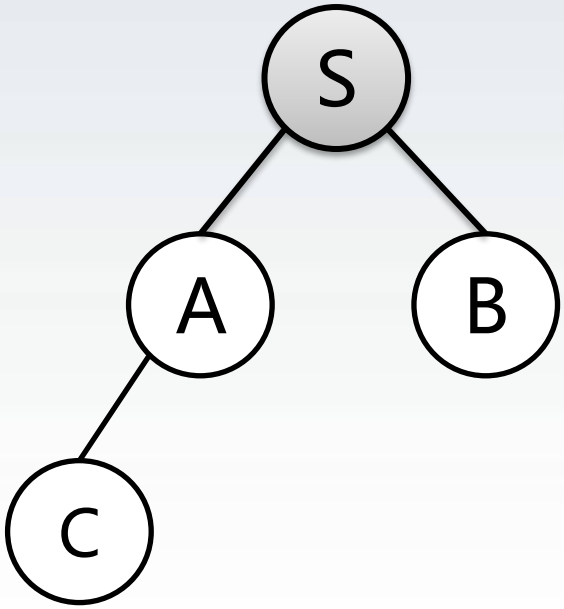
一个队列为什么不行？

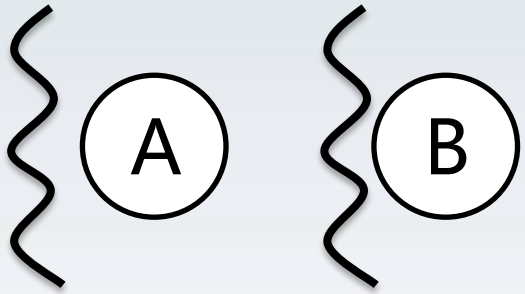




图广度优先搜索算法的并行化方法研究

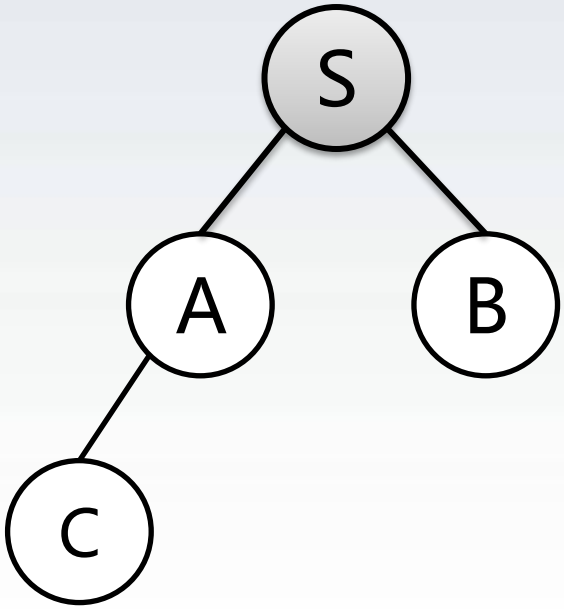
一个队列为什么不行？

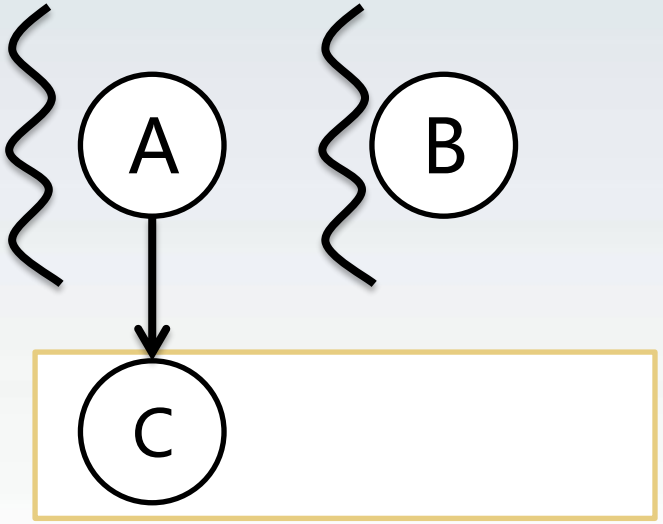




图广度优先搜索算法的并行化方法研究

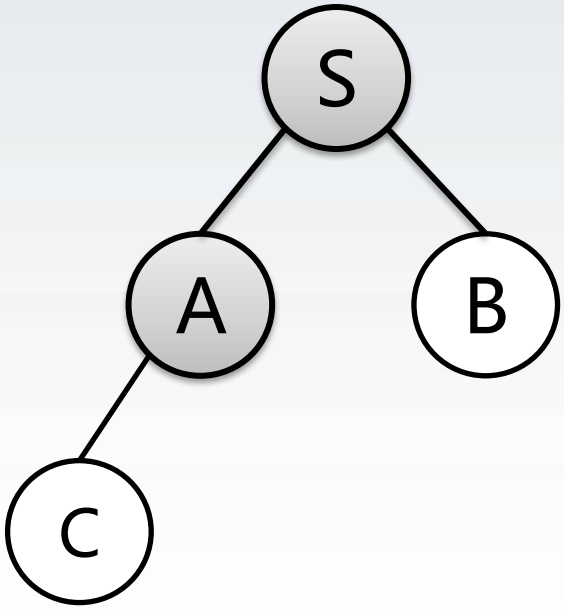
一个队列为什么不行？

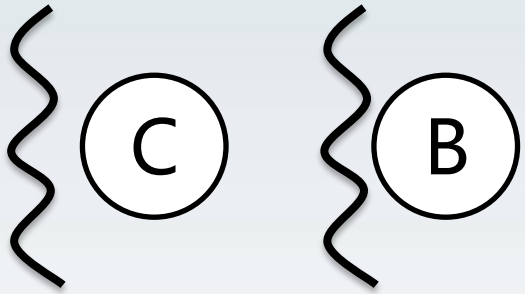




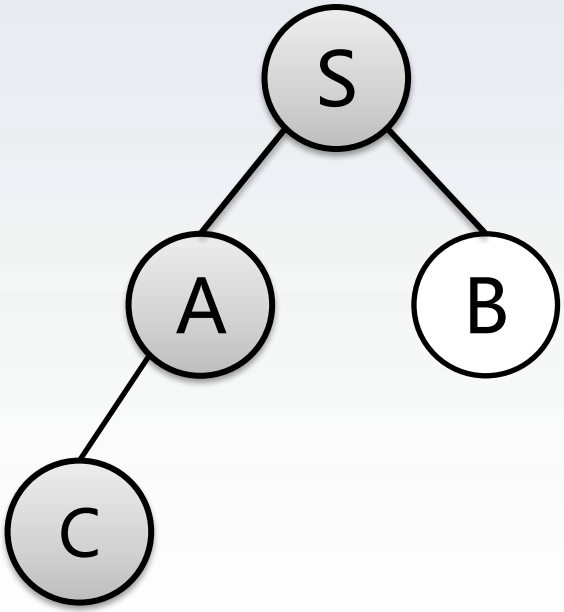
图广度优先搜索算法的并行化方法研究

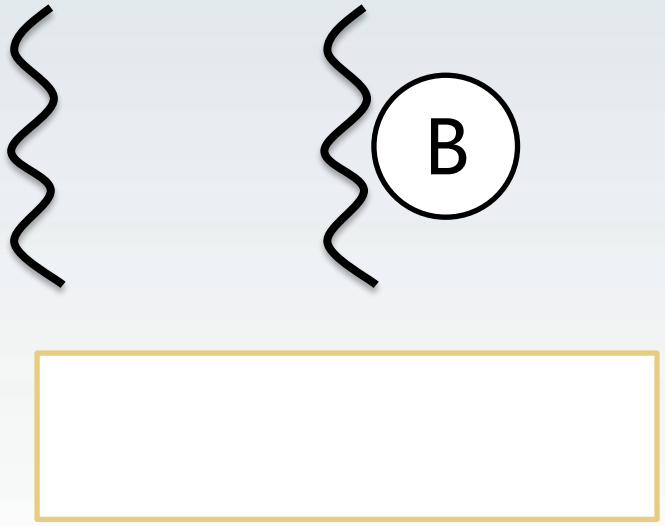
—个队列为什么不行？





一个队列为什么不行？





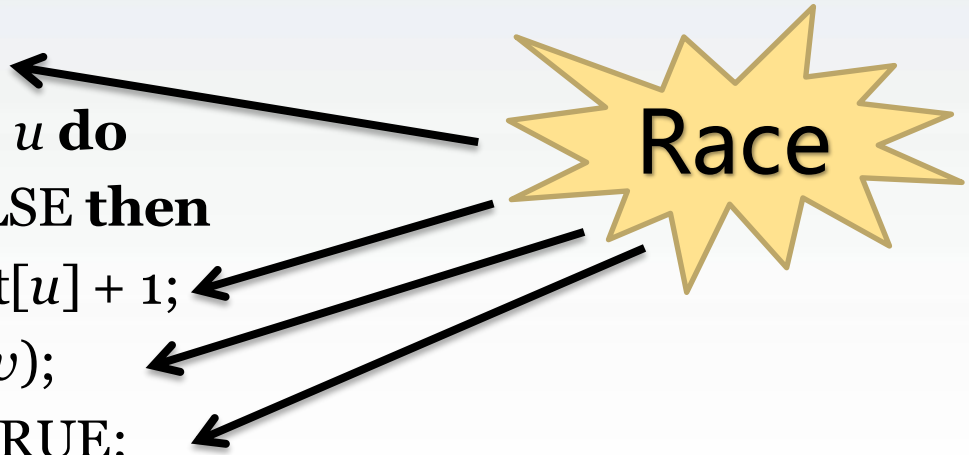
在访问第k层前访问了k+1层！

并行BFS算法

```
1.  fork;
2.  while CQ ≠ Ø do
3.      NQ ← Ø;
4.      while CQ ≠ Ø in parallel do
5.          u ← Dequeue(CQ);
6.          for each v adjacent to u do
7.              if Visited[v] = FALSE then
8.                  Cost[v] ← Cost[u] + 1;
9.                  Enqueue(NQ, v);
10.                 Visited[v] ← TRUE;
11.     Swap(CQ, NQ);
12. join;
```

并行BFS算法

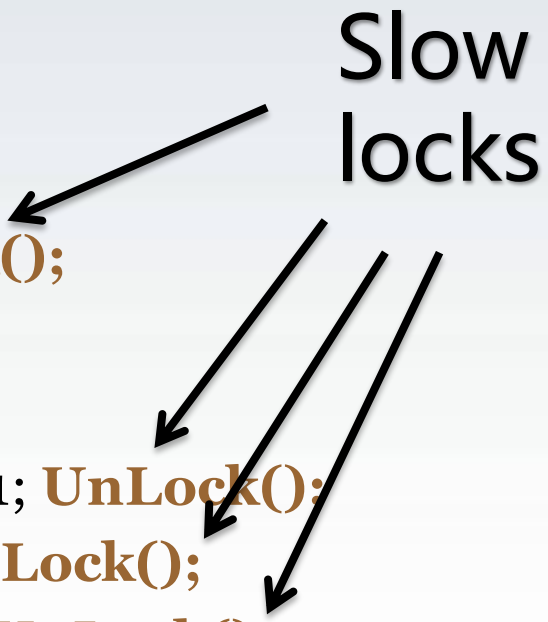
```
1.  fork;
2.  while CQ ≠ Ø do
3.      NQ ← Ø;
4.      while CQ ≠ Ø in parallel do
5.          u ← Dequeue(CQ);
6.          for each v adjacent to u do
7.              if Visited[v] = FALSE then
8.                  Cost[v] ← Cost[u] + 1;
9.                  Enqueue(NQ, v);
10.                 Visited[v] ← TRUE;
11.     Swap(CQ, NQ);
12. join;
```



并行BFS算法

锁风格

```
1.  fork;
2.  while CQ ≠ Ø do
3.      NQ ← Ø;
4.      while CQ ≠ Ø in parallel do
5.          Lock();u ← Dequeue(CQ); UnLock();
6.          for each v adjacent to u do
7.              if Visited[v] = FALSE then
8.                  Lock(); Cost[v] ← Cost[u] + 1; UnLock();
9.                  Lock(); Enqueue(NQ, v); UnLock();
10.                 Lock(); Visited[v] ← TRUE; UnLock();
11.     Swap(CQ, NQ);
12. join;
```



图广度优先搜索算法的并行化方法研究

并行BFS算法

无锁风格

```
1.  fork;
2.  while CQ ≠ Ø do
3.      NQ ← Ø;
4.      while CQ ≠ Ø in parallel do
5.          u ← AtomicDequeue(CQ);
6.          for each v adjacent to u do
7.              old = ReadAndSet(Visted[v], True);
8.              if old= FALSE then
9.                  Cost[v] ← Cost[u] + 1;
10.                 AtomicEnqueue(NQ, v);
11.     Swap(CQ, NQ);
12. join;
```

原子操作

并行BFS算法

原子操作比锁好在哪?

- **Lock();** a+=1; **Unlock();**

  多条指令

  获取锁的汇编代码

```
mov    eax, dword ptr [a]
add    eax, 1
mov    dword ptr [a], eax
```

  释放锁的汇编代码

- GCC: **__sync_fetch_and_add(**&a, 1**);**

  一条指令

```
lock xadd dword ptr[ecx], eax
```

并行BFS算法

考虑线程同步

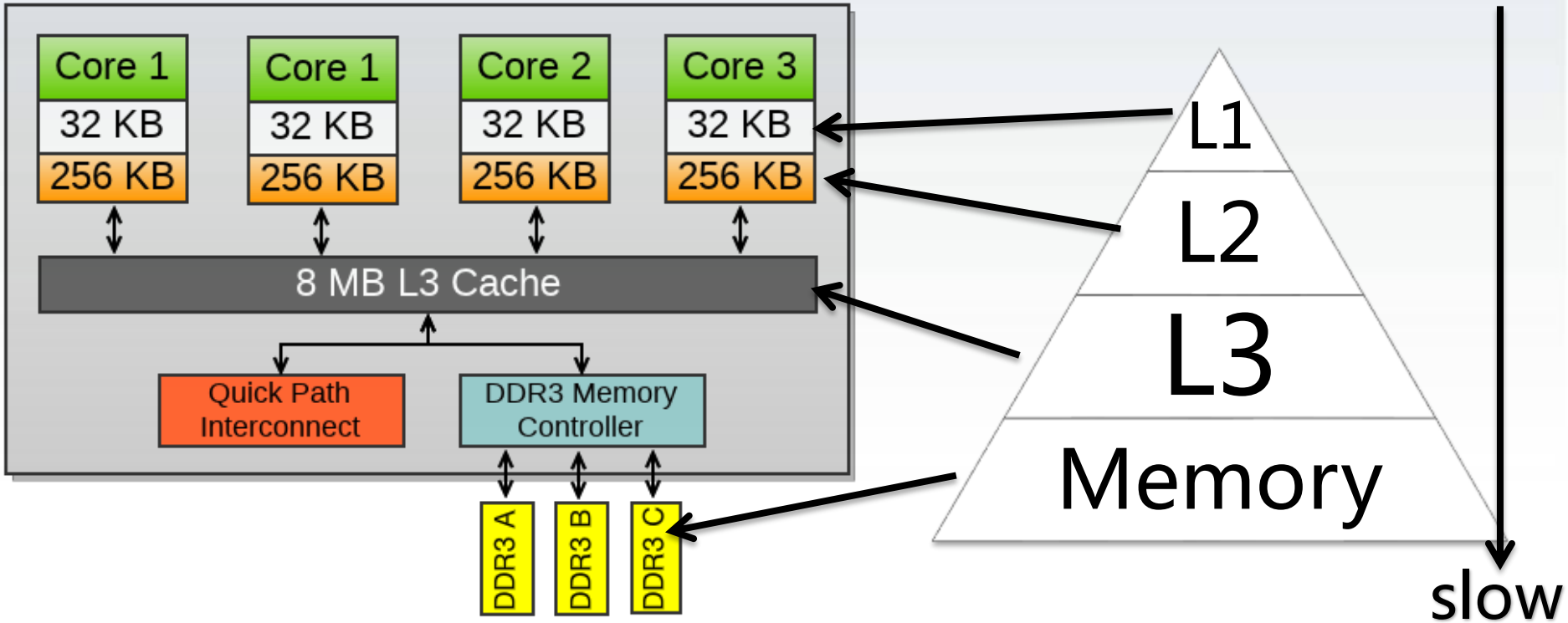
```
1.  fork;
2.  while CQ ≠ Ø do
3.      NQ ← Ø;
4.      while CQ ≠ Ø in parallel do
5.          u ← AtomicDequeue(CQ);
6.          for each v adjacent to u do
7.              old = ReadAndSet(Visted[v], True);
8.              if old= FALSE then
9.                  Cost[v] ← Cost[u] + 1;
10.                 AtomicEnqueue(NQ, v);
11.     Swap(CQ, NQ);
12. join;
```

图广度优先搜索算法的并行化方法研究

介绍 | BFS | 优化 | 实验 | 总结

- CPU优化
  - 位图

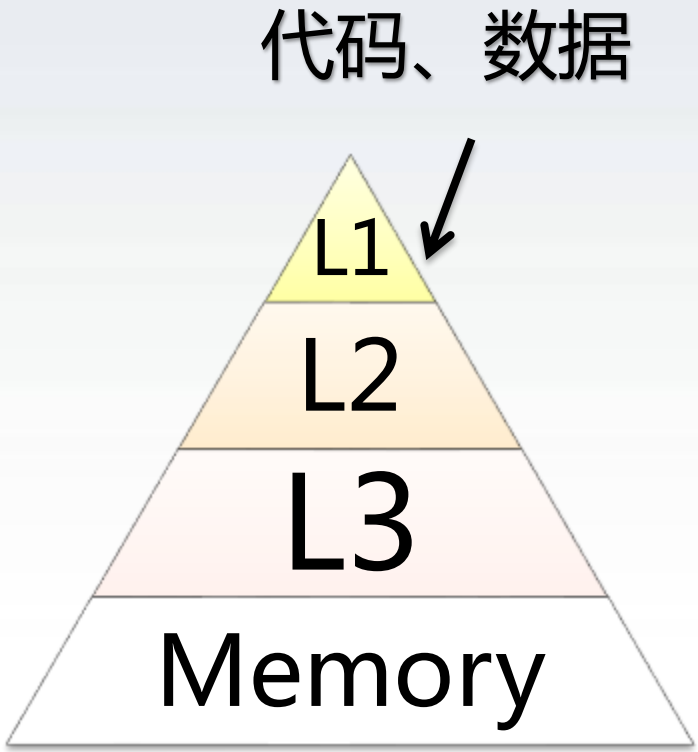
CPU的多级存储结构



图广度优先搜索算法的并行化方法研究

- CPU优化
  - 位图

程序的局部性原理



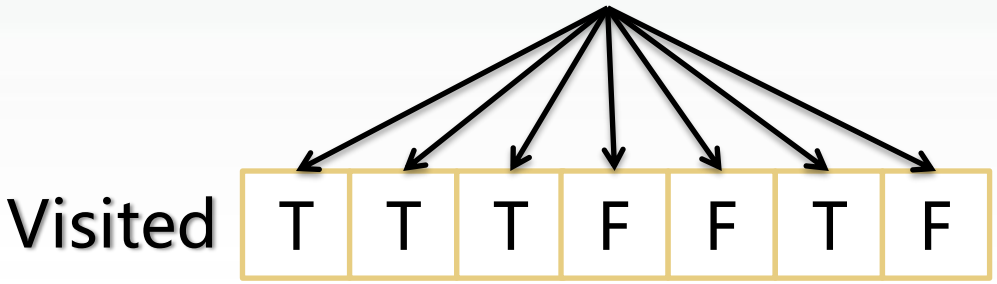
图广度优先搜索算法的并行化方法研究

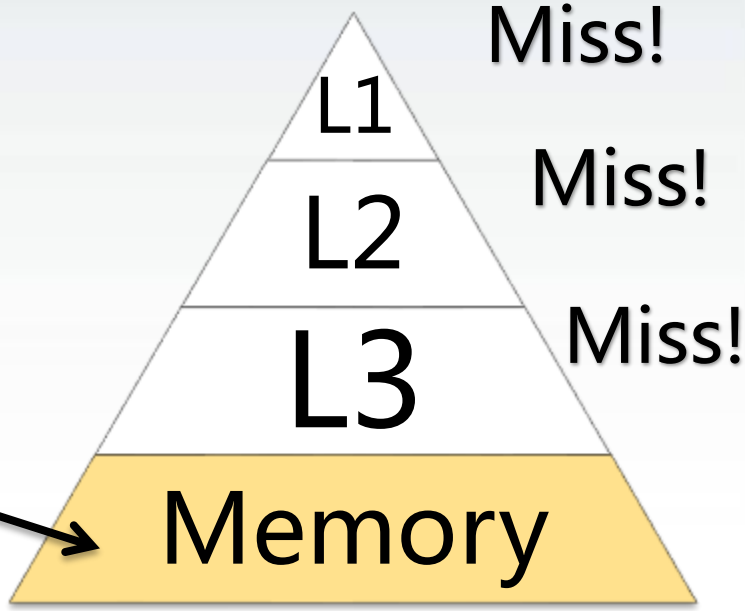
介绍 | BFS | 优化 | 实验 | 总结

- CPU优化
  - 位图

程序的局部性原理

BFS随机访问



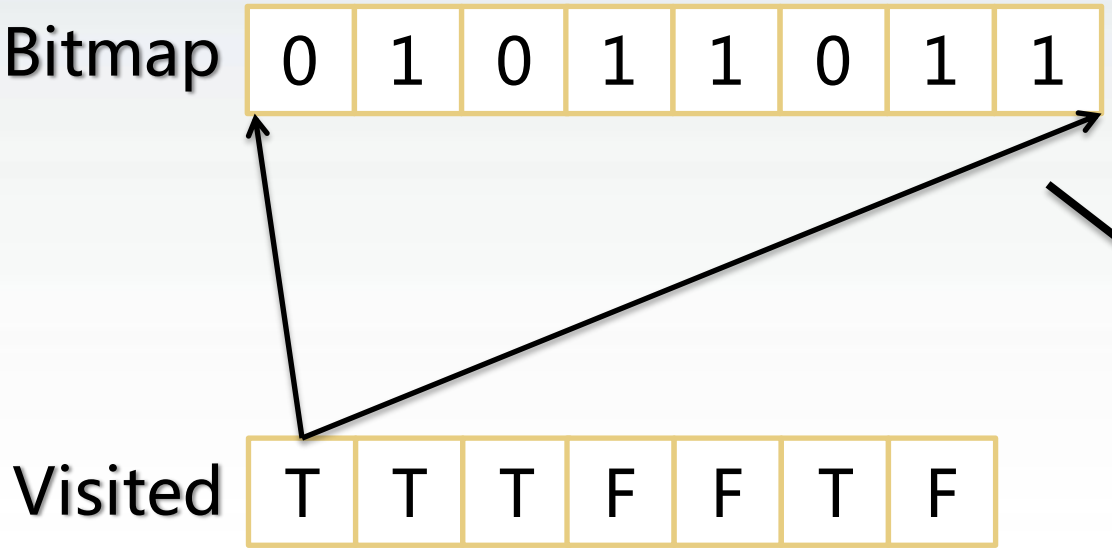


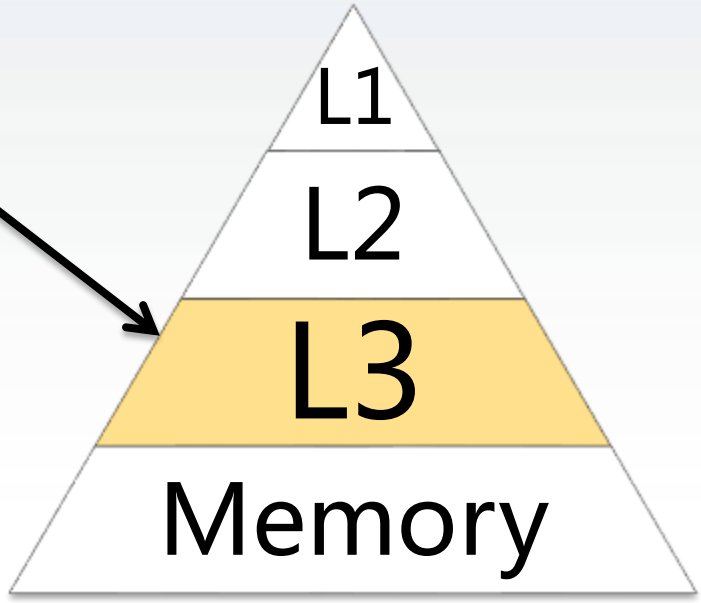
图广度优先搜索算法的并行化方法研究

介绍 | BFS | 优化 | 实验 | 总结

- CPU优化
  - 位图

用bool变量的8个bit位来表示

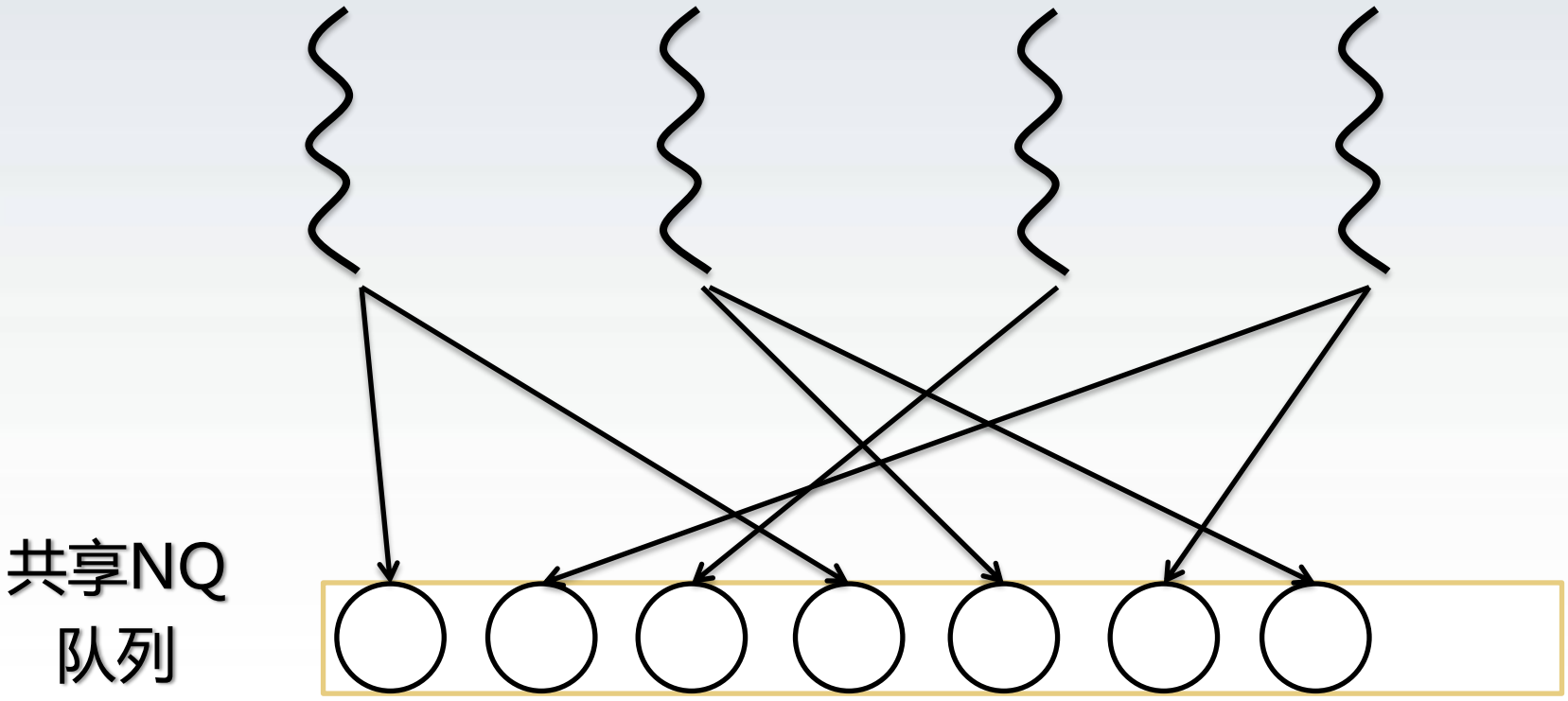




图广度优先搜索算法的并行化方法研究

介绍 | BFS | 优化 | 实验 | 总结

- CPU优化
  - 两级队列



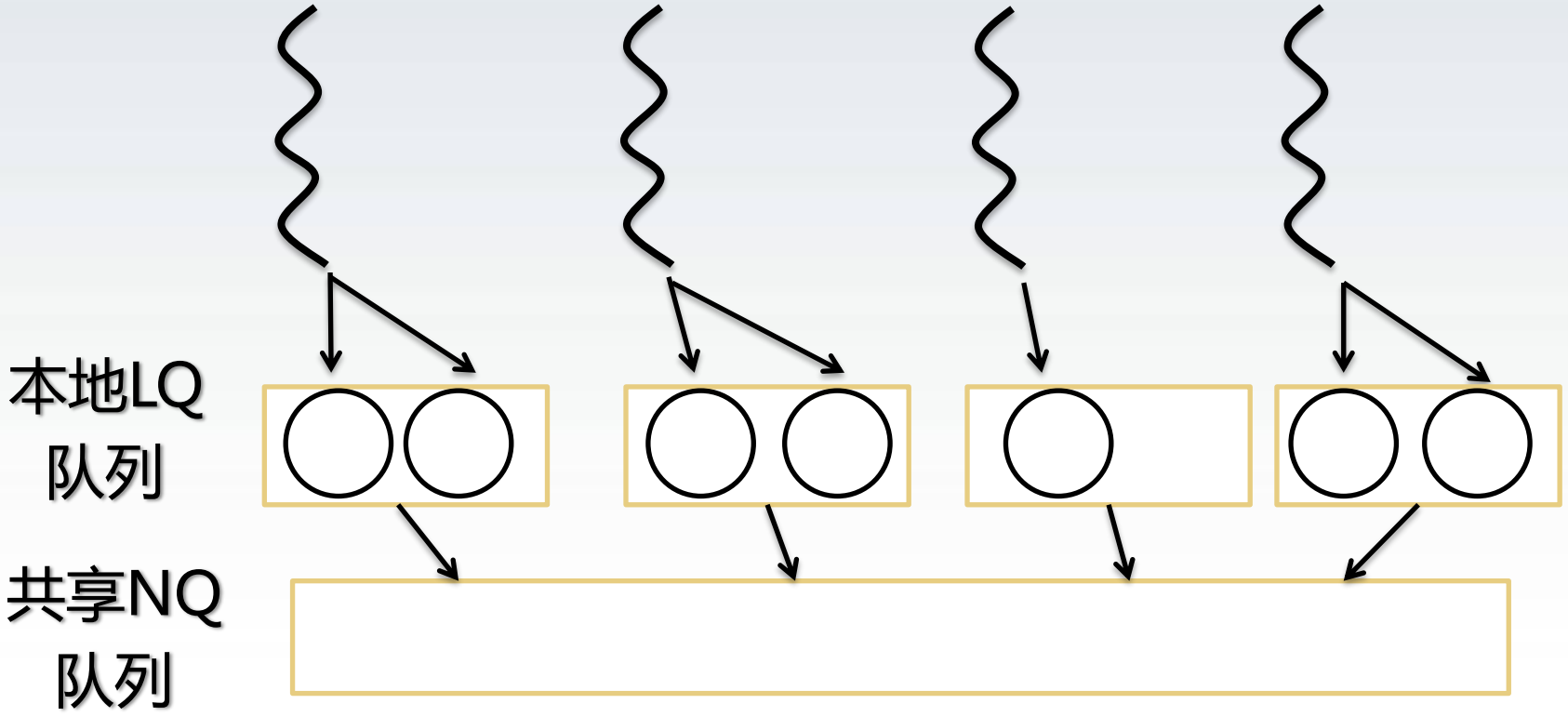
大量原子操作导致绝大多数线程停滞

图广度优先搜索算法的并行化方法研究

介绍 | BFS | 优化 | 实验 | 总结

CPU优化

两级队列

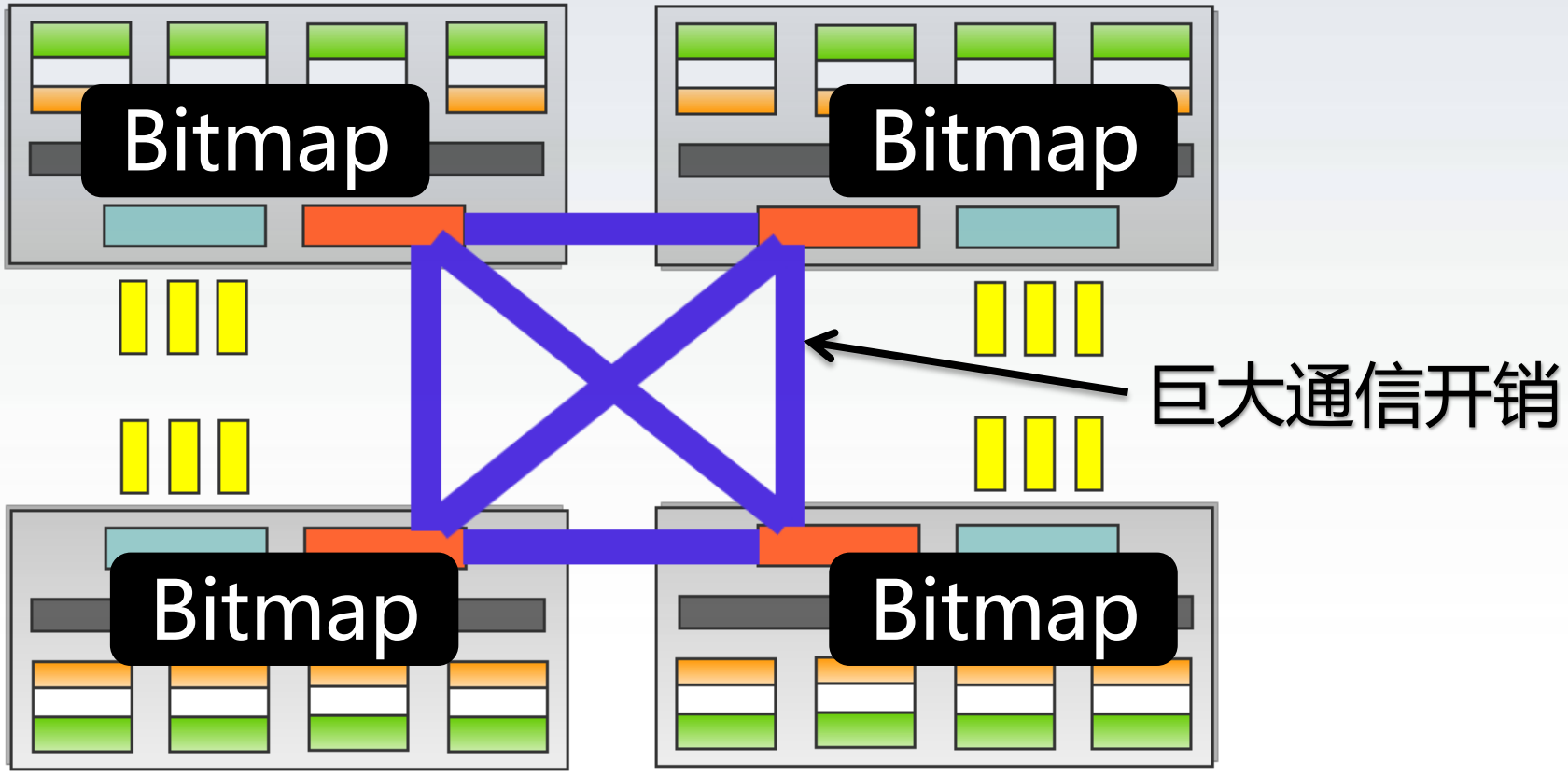


使用局部队列将顶点先缓存起来

图广度优先搜索算法的并行化方法研究

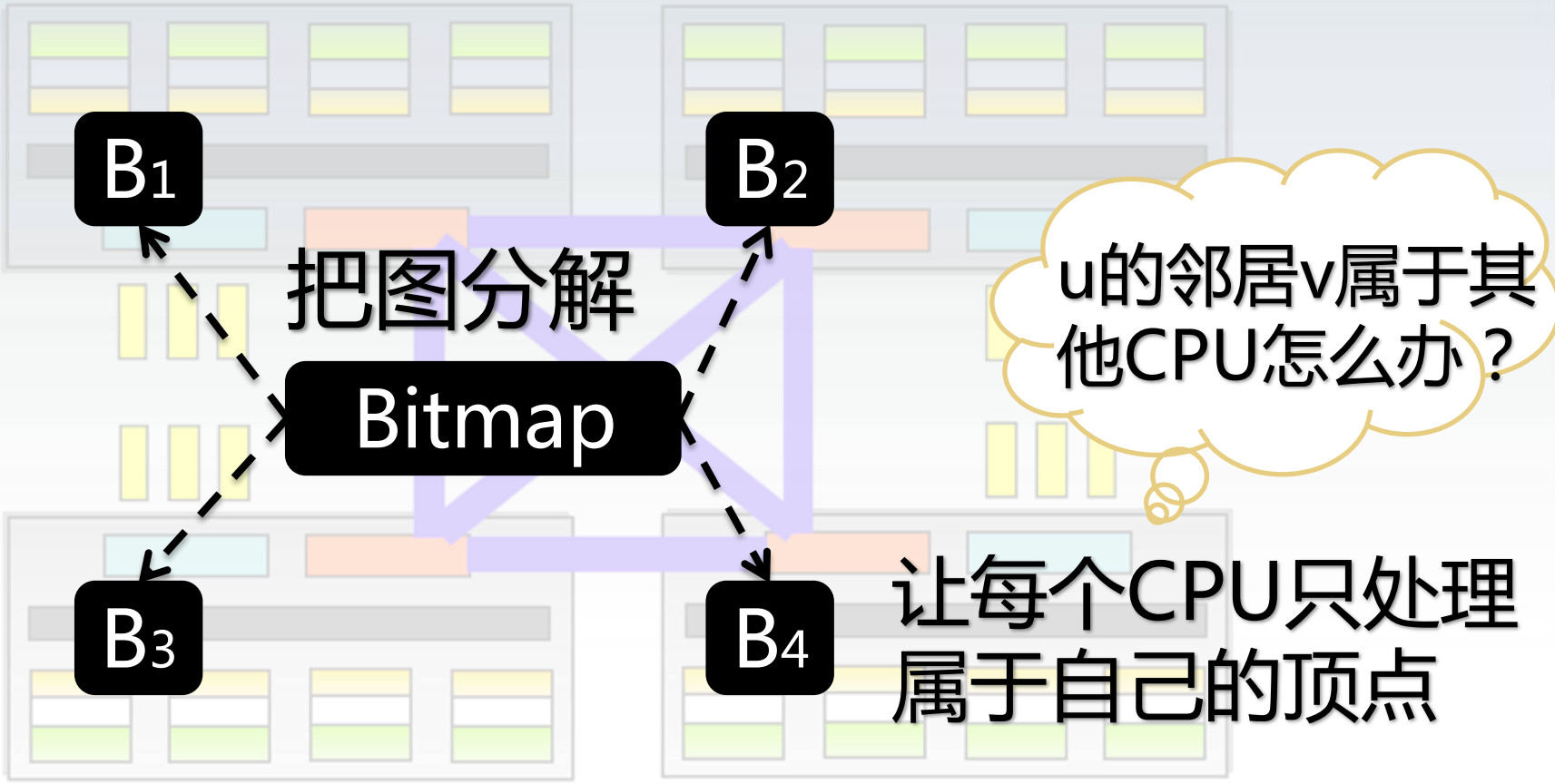
介绍 | BFS | 优化 | 实验 | 总结

- CPU优化
  - Socket队列



图广度优先搜索算法的并行化方法研究

- CPU优化
  - Socket队列

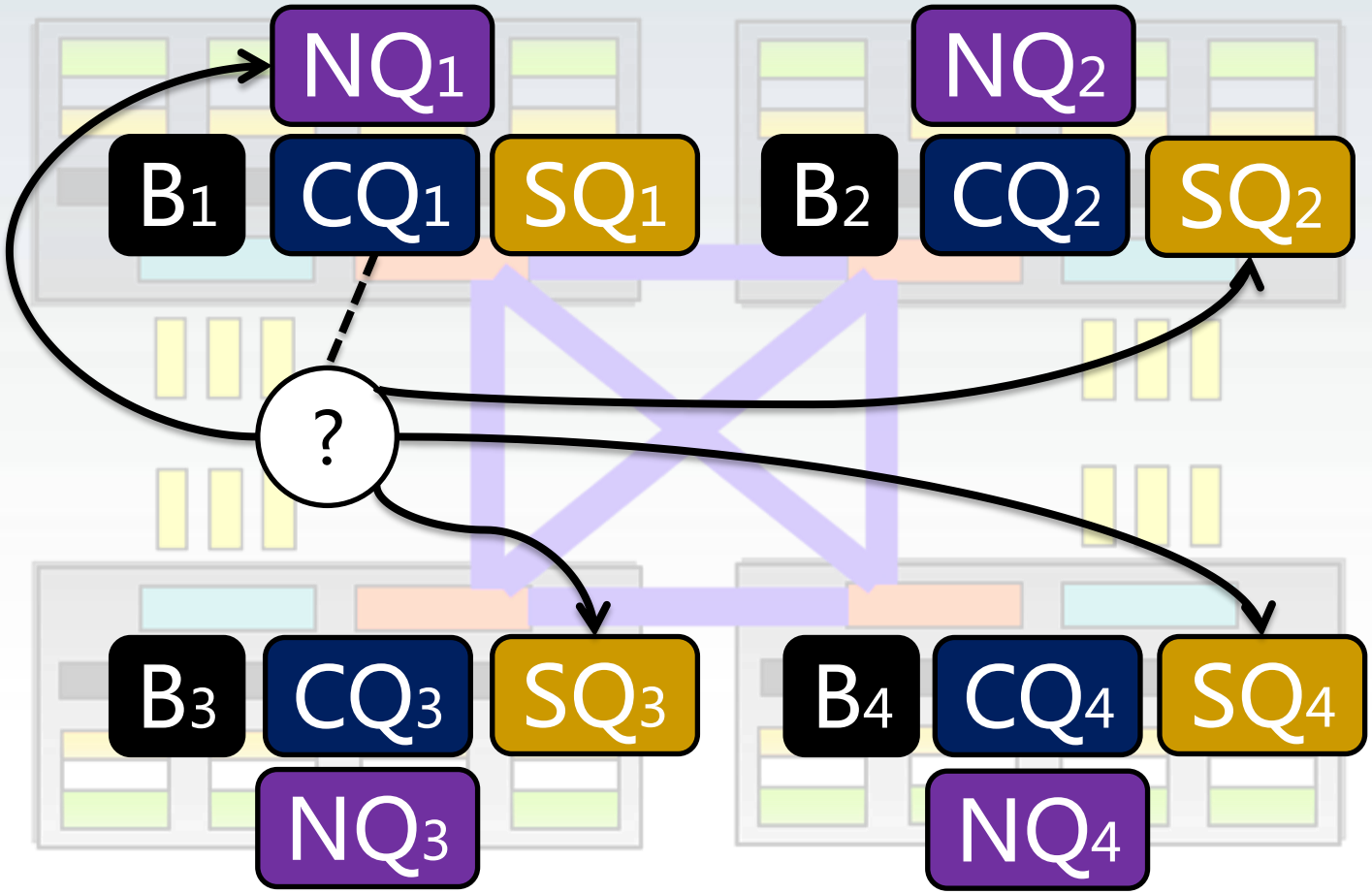


图广度优先搜索算法的并行化方法研究

介绍 | BFS | 优化 | 实验 | 总结

CPU优化

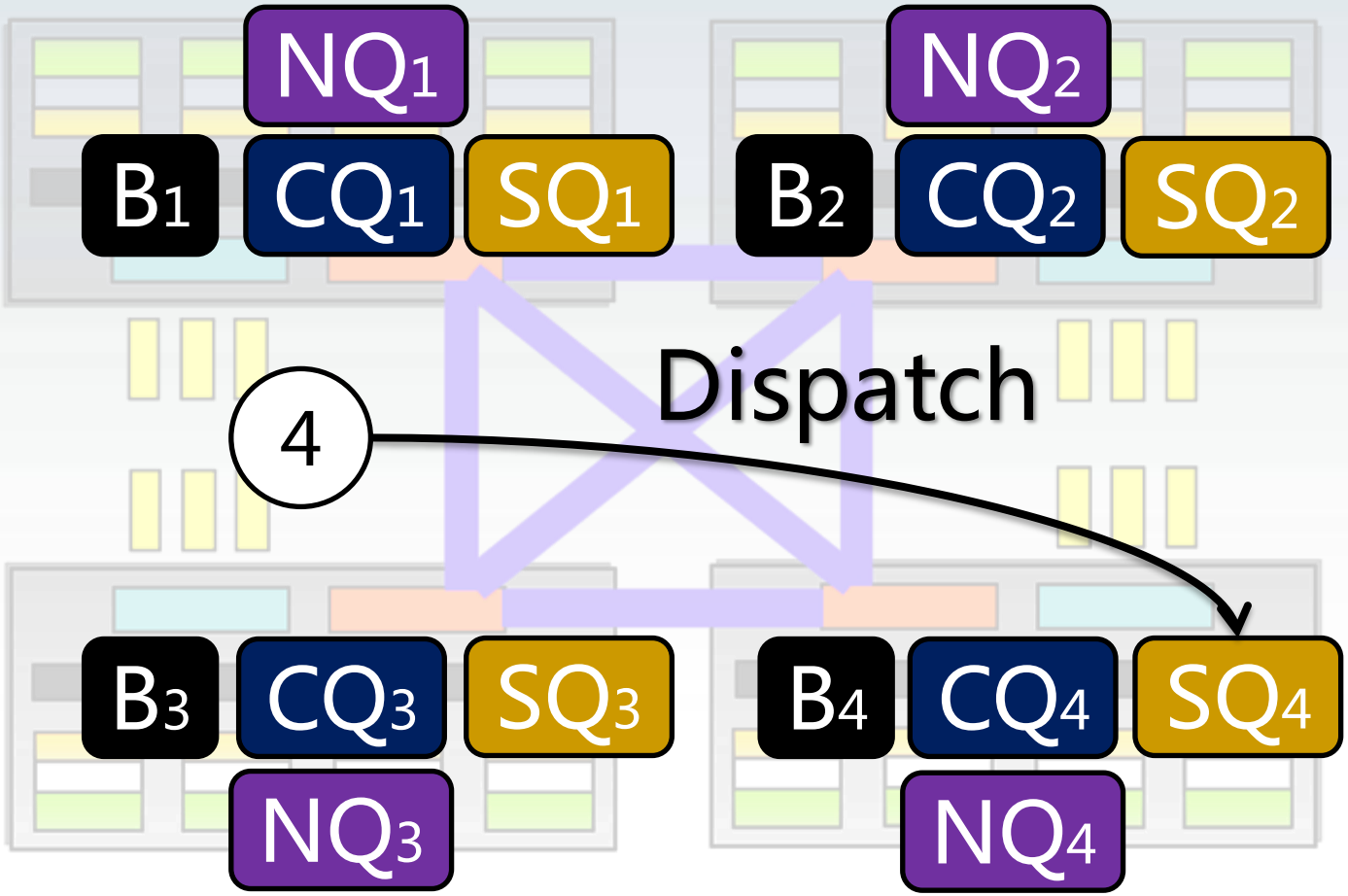
Socket队列



图广度优先搜索算法的并行化方法研究

介绍 | BFS | 优化 | 实验 | 总结

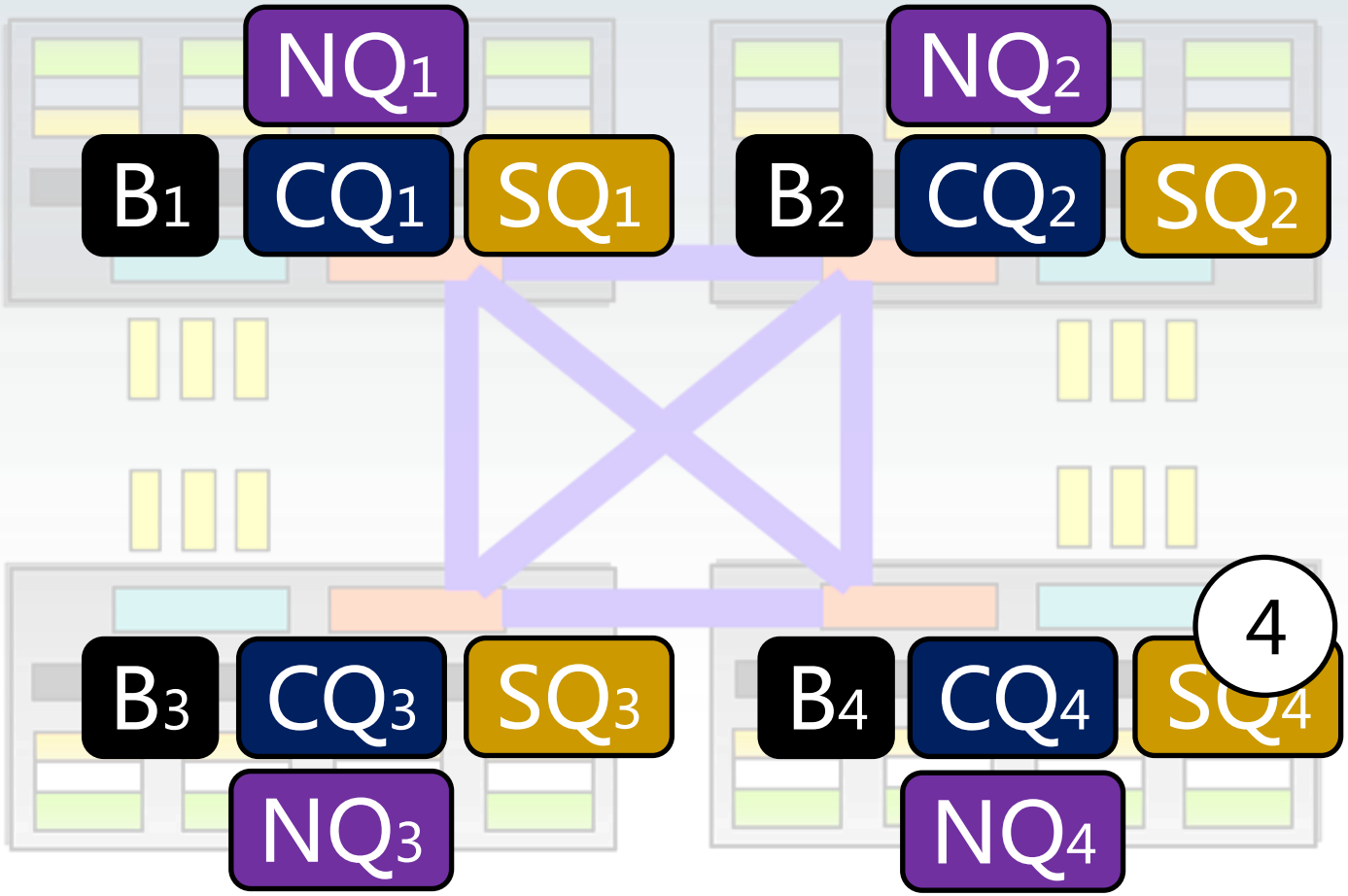
- CPU优化
  - Socket队列



图广度优先搜索算法的并行化方法研究

介绍 | BFS | 优化 | 实验 | 总结

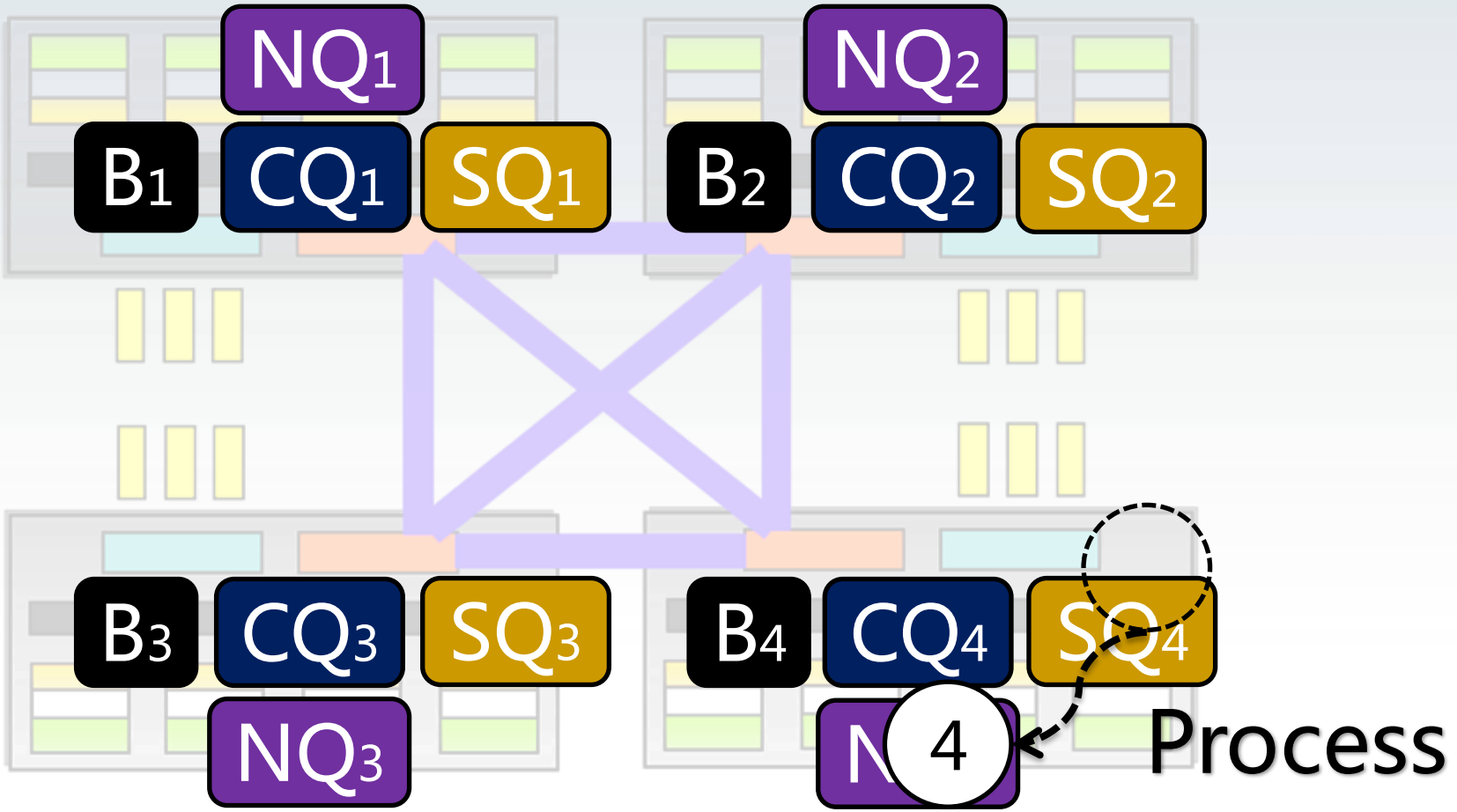
- CPU优化
  - Socket队列



图广度优先搜索算法的并行化方法研究

介绍 | BFS | 优化 | 实验 | 总结

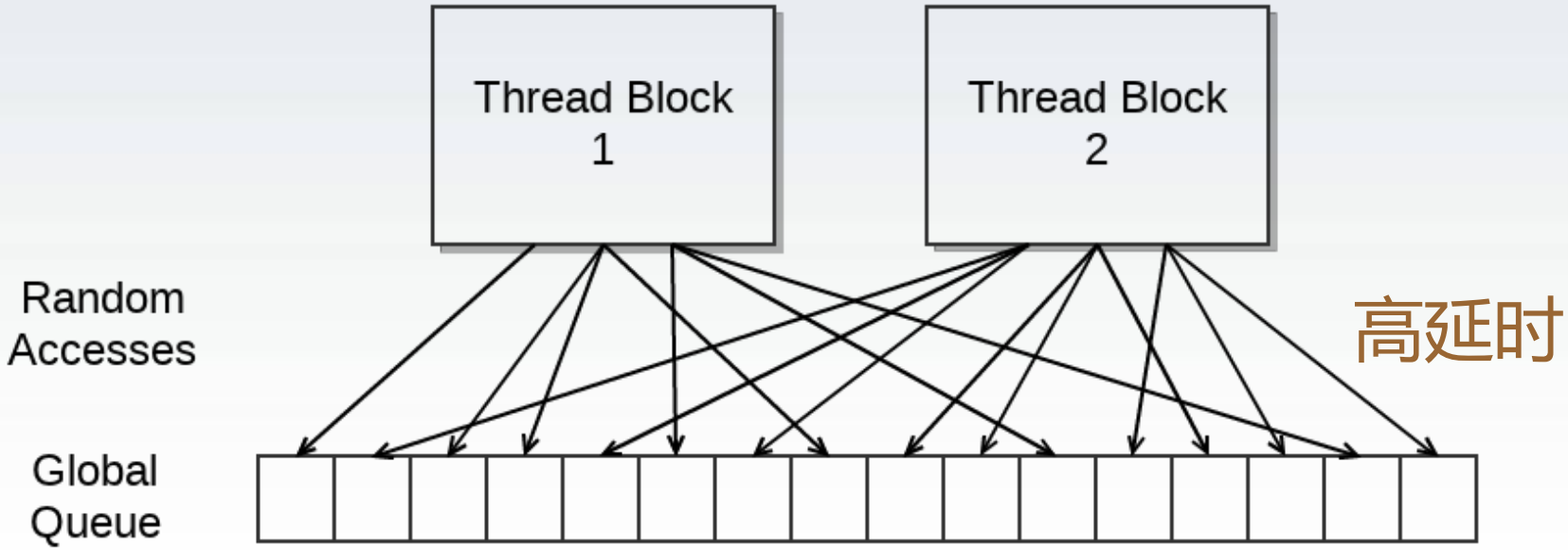
- CPU优化
  - Socket队列



图广度优先搜索算法的并行化方法研究

GPU优化

两级队列



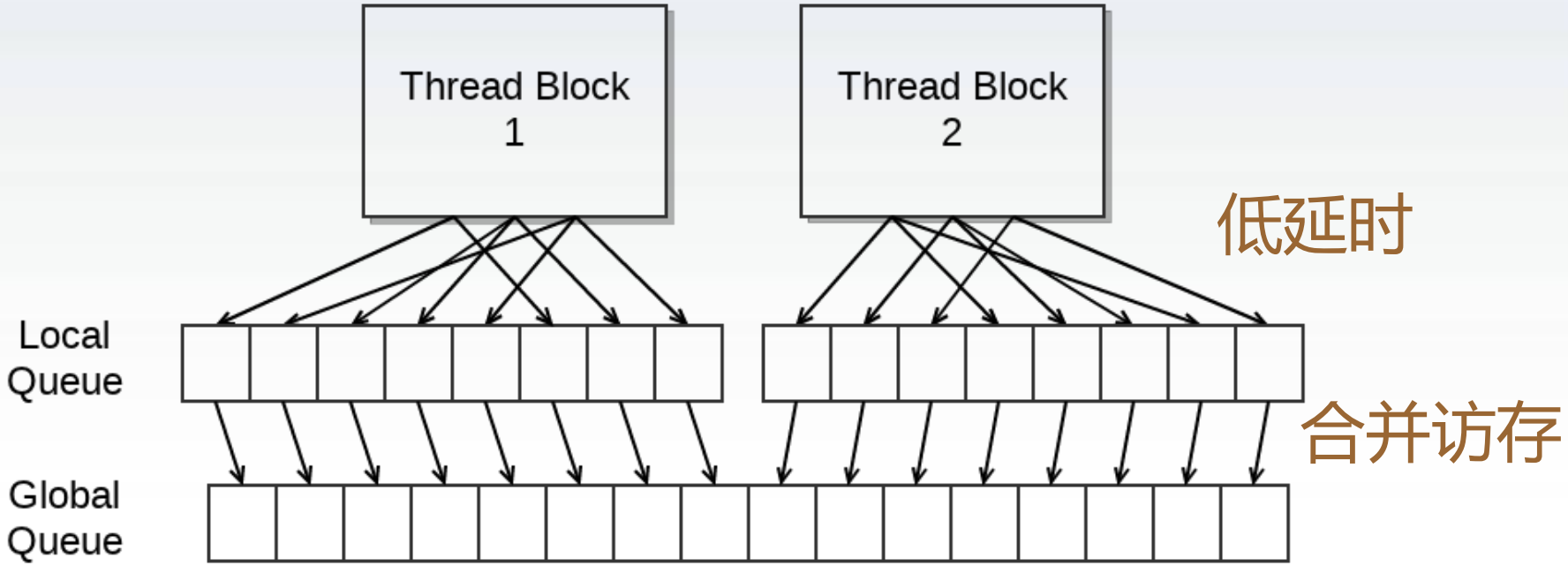
问题：存在大量的不规则访存

图广度优先搜索算法的并行化方法研究

介绍 | BFS | 优化 | 实验 | 总结

GPU优化

两级队列



使用片内存储，使对显存的访问变得规整

- GPU优化
  - 预读

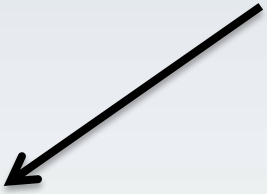
for each v adjacent to u **do**

```
old = ReadAndSet(Visted[v], True);
if old= FALSE then
    Cost[v] ← Cost[u] + 1;
    AtomaticEnqueue(NQ, v);
```

图广度优先搜索算法的并行化方法研究

- GPU优化
  - 预读

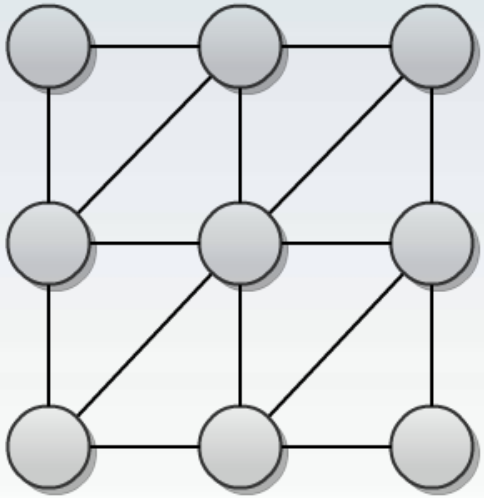
过滤大量原子操作



```
for each v adjacent to u do
    if Visited[v] == False then
        old = ReadAndSet(Visted[v], True);
        if old= FALSE then
            Cost[v] ← Cost[u] + 1;
            AtomaticEnqueue(NQ, v);
```

图广度优先搜索算法的并行化方法研究

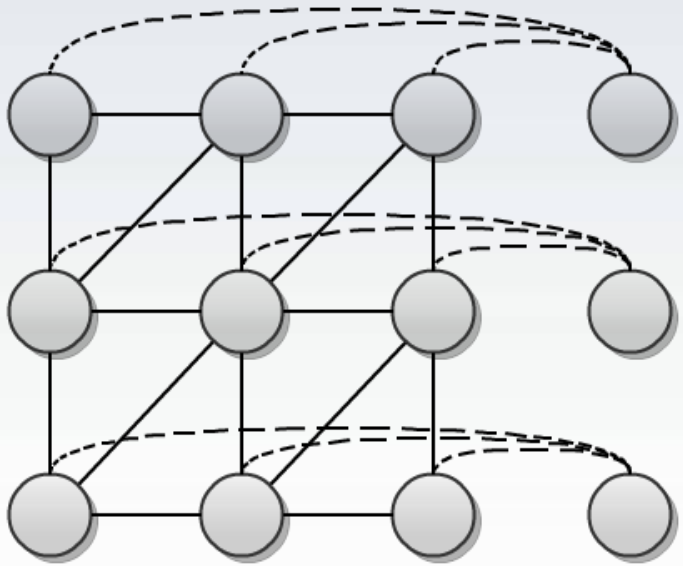
- 用例
  - 规则图



Grid

图广度优先搜索算法的并行化方法研究

- 用例
  - 不规则图



Isolated Nodes

用例

小世界网络

- 社交网络、互联网节点
- 使用SNAP库生成

  Stanford **N**etwork **A**nalysis **P**roject
- 平均直径短（7到11）
- 广度优先树的层数少

用例

真实世界网络

- 全美道路交通网图

  From 9th DIMACS Implementation Challenge - Shortest Paths

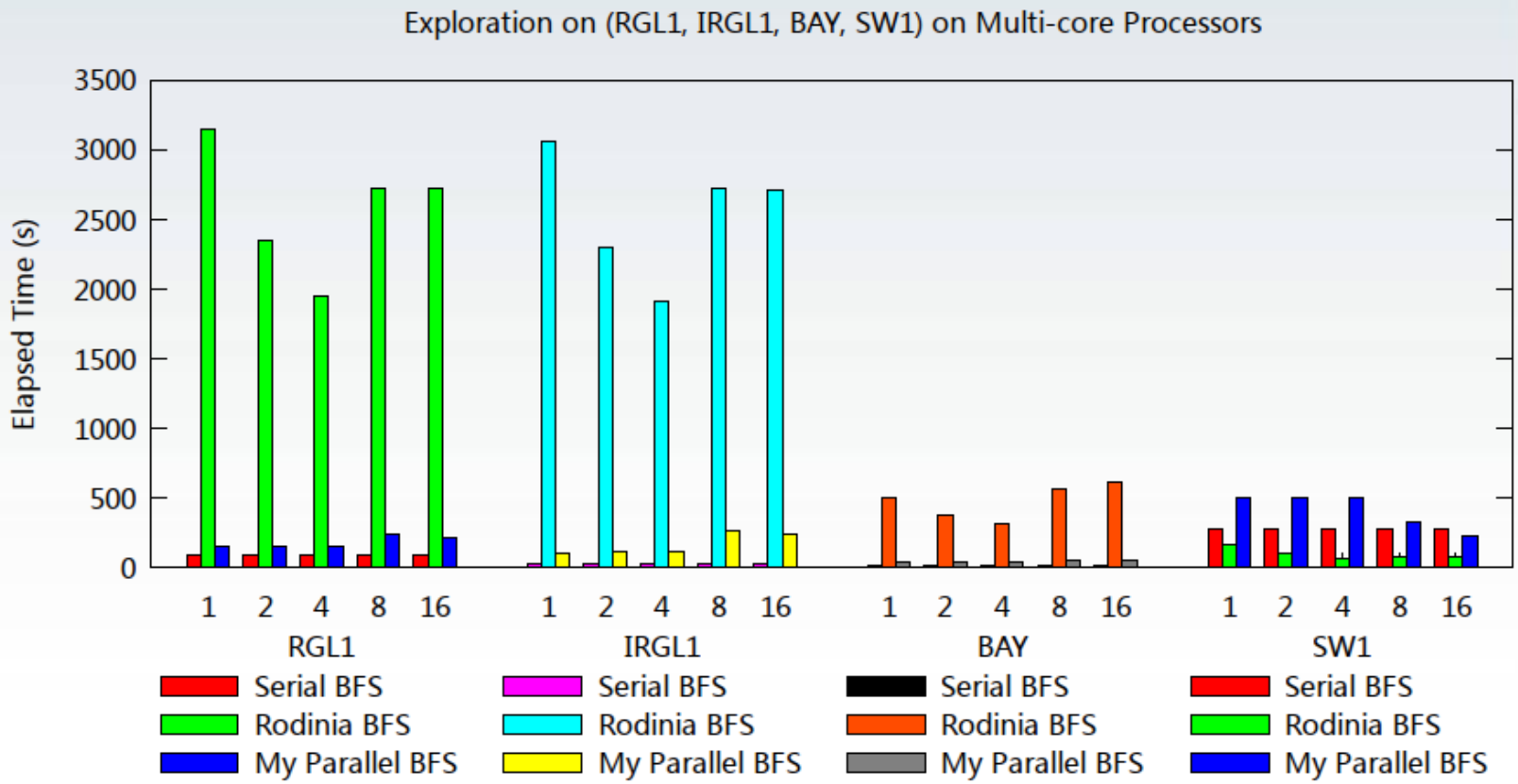
- 稀松、平均度数低（2、3）

图广度优先搜索算法的并行化方法研究

介绍 | BFS | 优化 | 实验 | 总结

结果

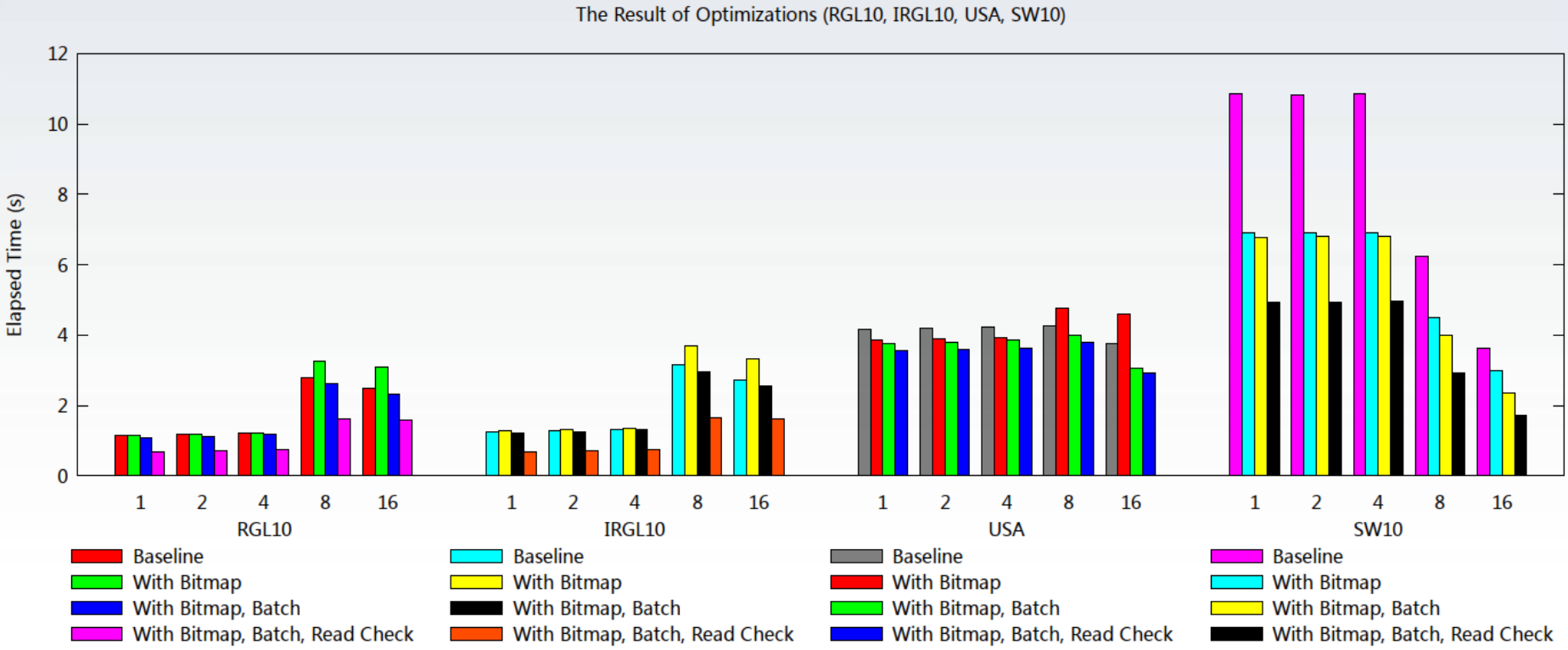
CPU基线程序



图广度优先搜索算法的并行化方法研究

介绍 | BFS | 优化 | 实验 | 总结

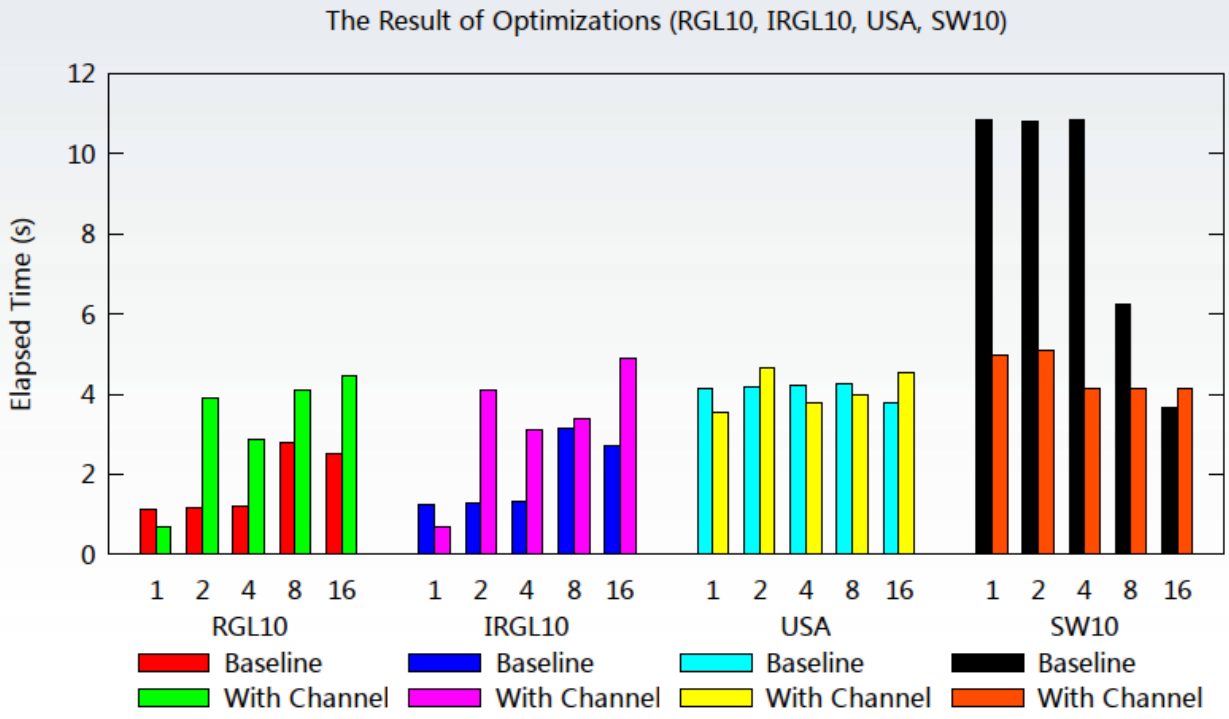
- 结果
  - CPU优化
    - 位图
    - 两级队列
    - 预读



图广度优先搜索算法的并行化方法研究

介绍 | BFS | 优化 | 实验 | 总结

- 结果
  - CPU优化
    - Socket队列



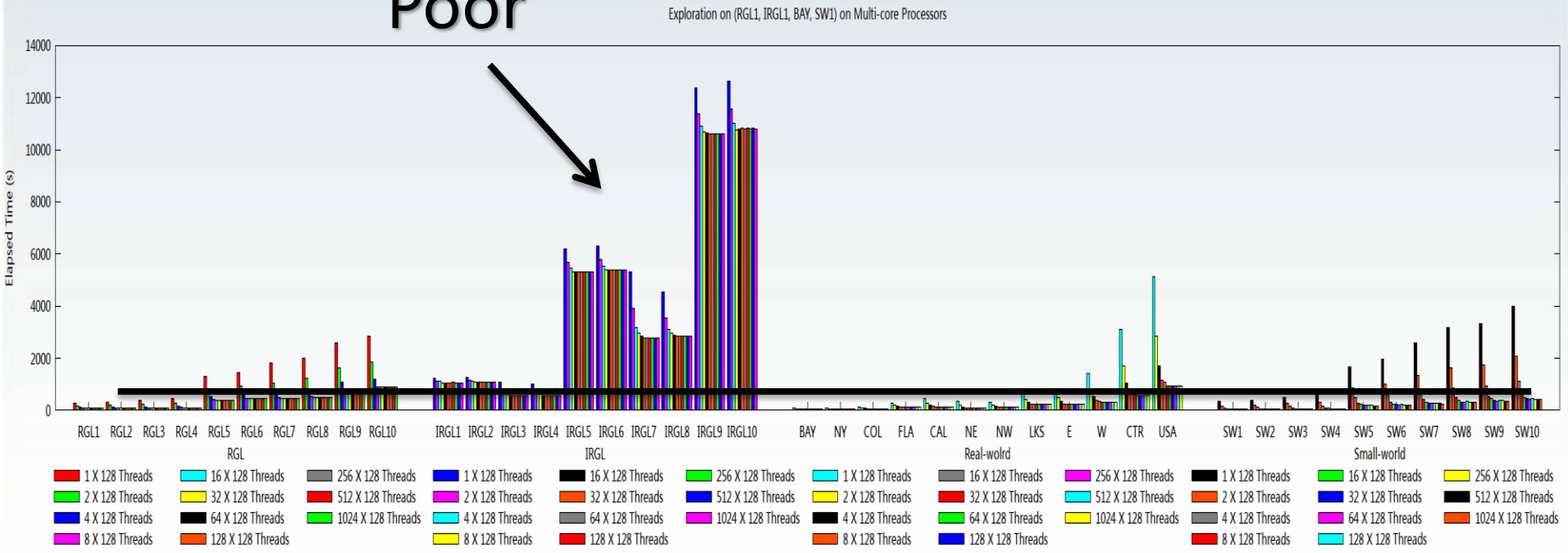
图广度优先搜索算法的并行化方法研究

介绍 | BFS | 优化 | 实验 | 总结

结果

GPU基线程序

Poor



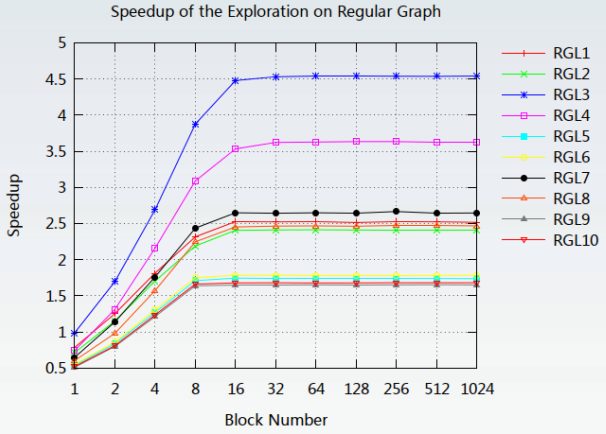
图广度优先搜索算法的并行化方法研究

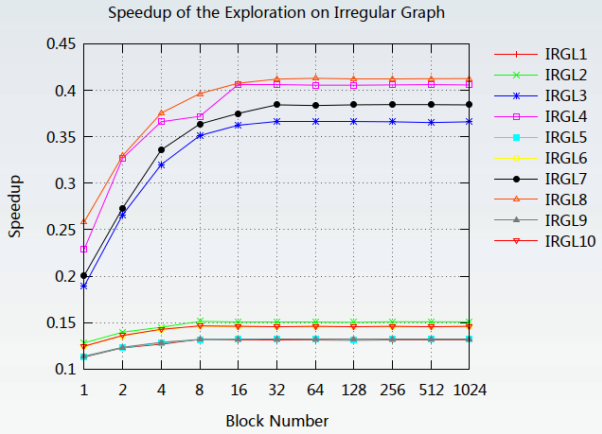
介绍 | BFS | 优化 | 实验 | 总结

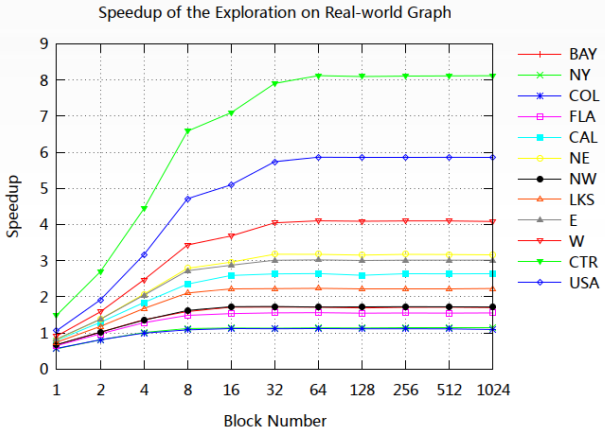
结果

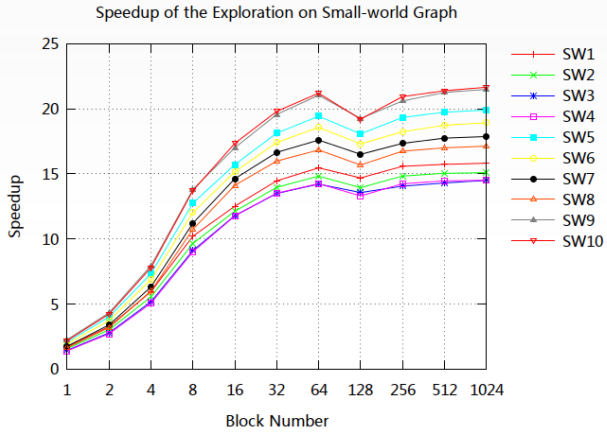
GPU基线程序

Speedup
规则: 1.7~4.5
不规则: 0.12~0.42
真实世界:1.1~8
小世界: 14~22









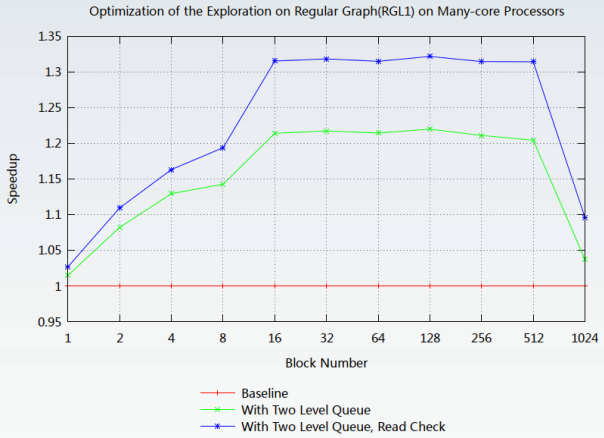
图广度优先搜索算法的并行化方法研究

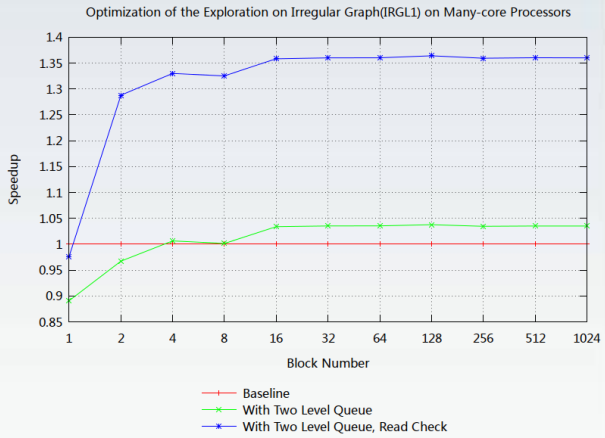
介绍 | BFS | 优化 | 实验 | 总结

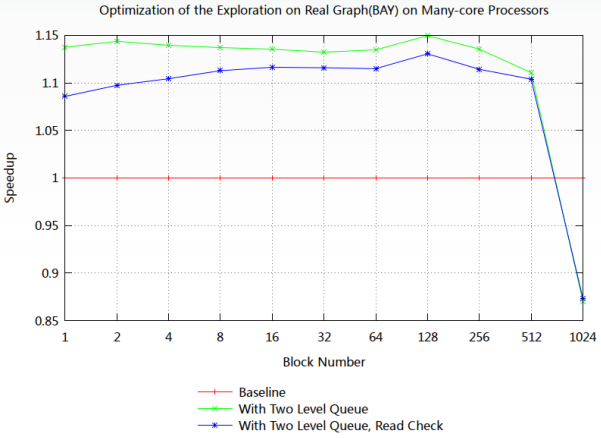
结果

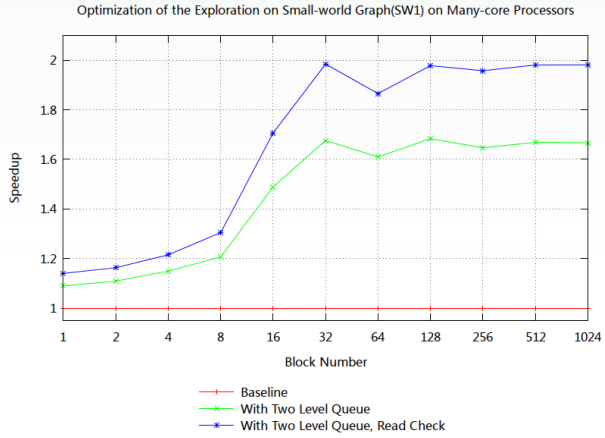
GPU优化

Speedup
规则: 1.32
不规则: 1.35
真实世界:1.15
小世界: 2



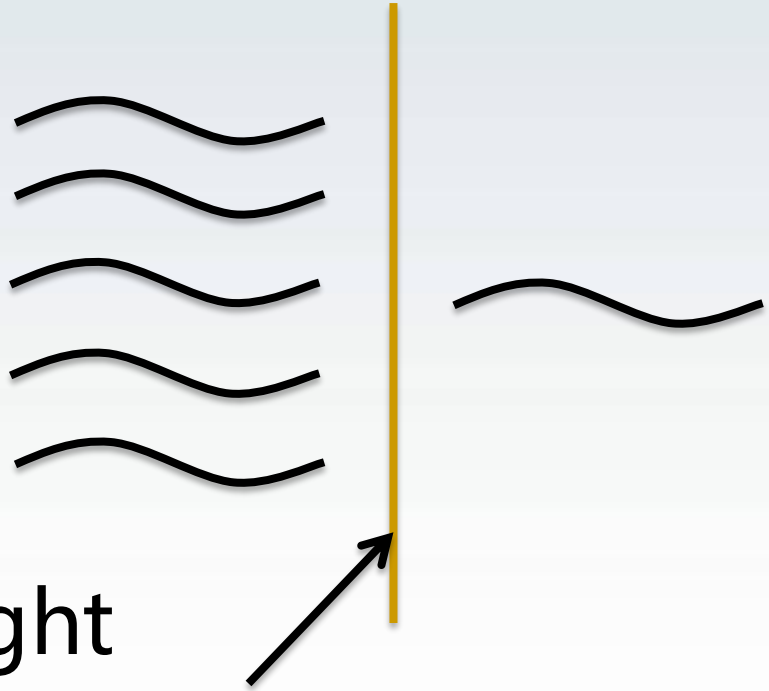




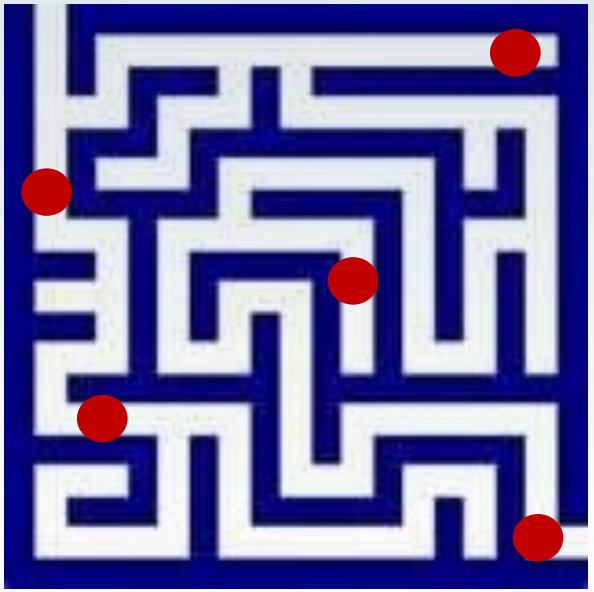


图广度优先搜索算法的并行化方法研究

为什么BFS很难并行？



Right
There
Waiting



Global Memory

图广度优先搜索算法的并行化方法研究

为什么GPU比CPU更适合并行BFS？

	CPU	GPU
设计目标	在更短延时内获取数据和指令	侧重数据的吞吐量
策略	逻辑控制、分支预测、多级缓存	大量执行单元相对简单线程
线程切换	软件	硬件
核心数量	16	448
强项	指令级并行	数据集并行

图广度优先搜索算法的并行化方法研究

介绍 | BFS | 优化 | 实验 | 总结

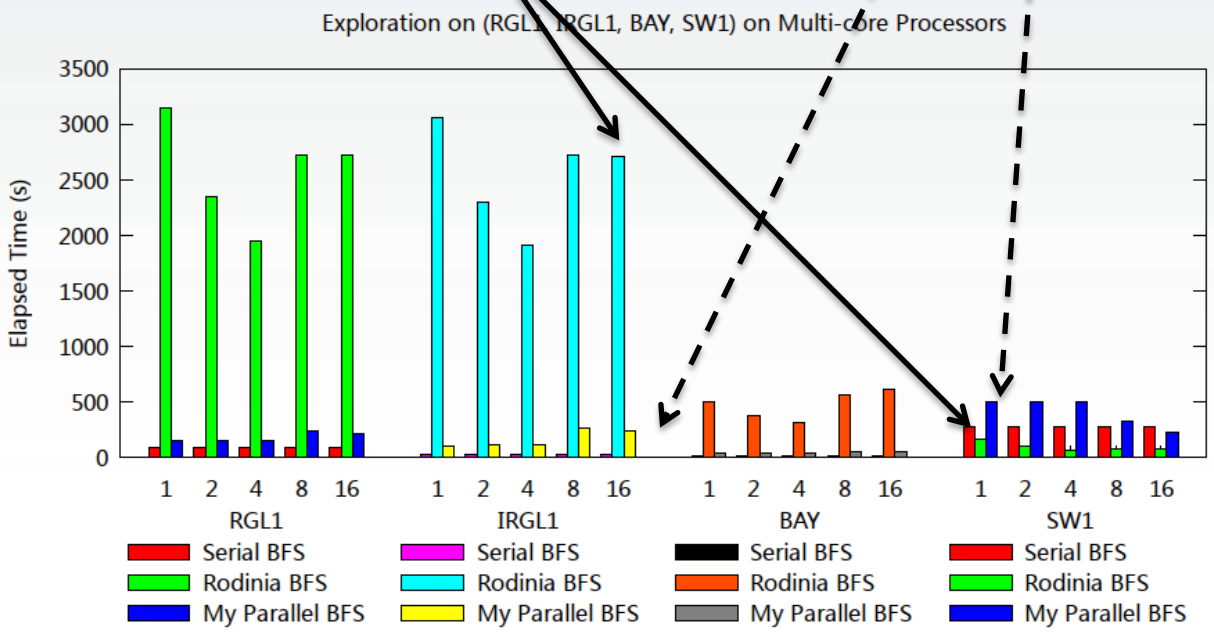
算法对并行BFS的影响有多大？

Rodinia

$O(V \cdot L + E)$

Mine

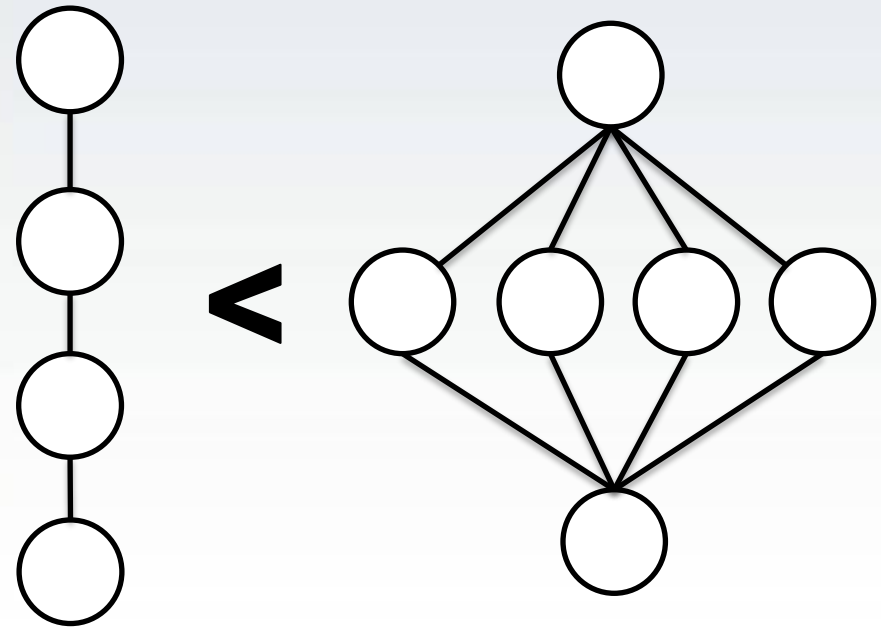
$O(V \cdot L + E)$

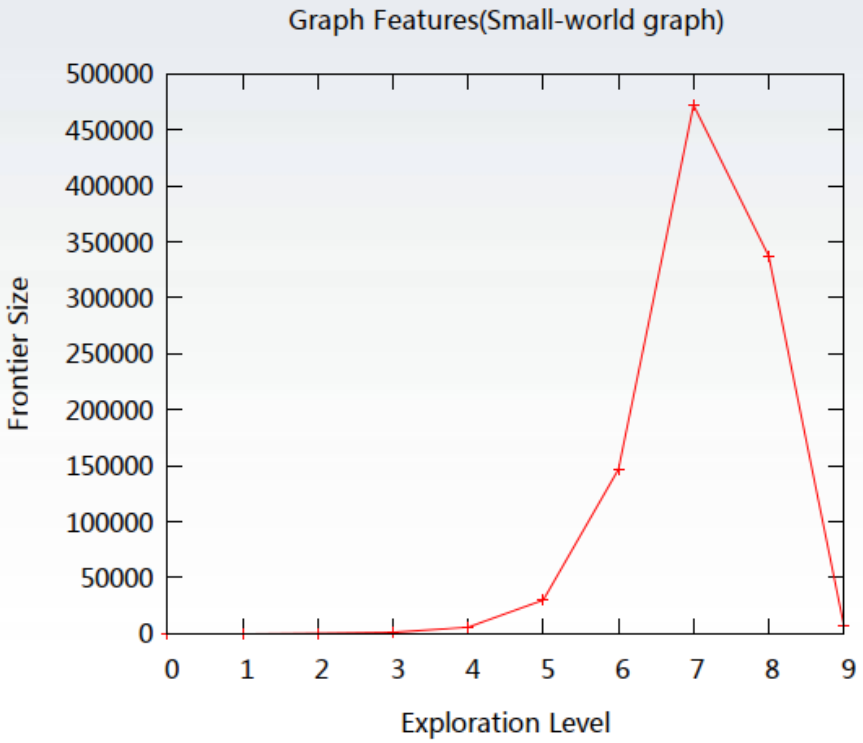


图广度优先搜索算法的并行化方法研究

介绍 | BFS | 优化 | 实验 | 总结

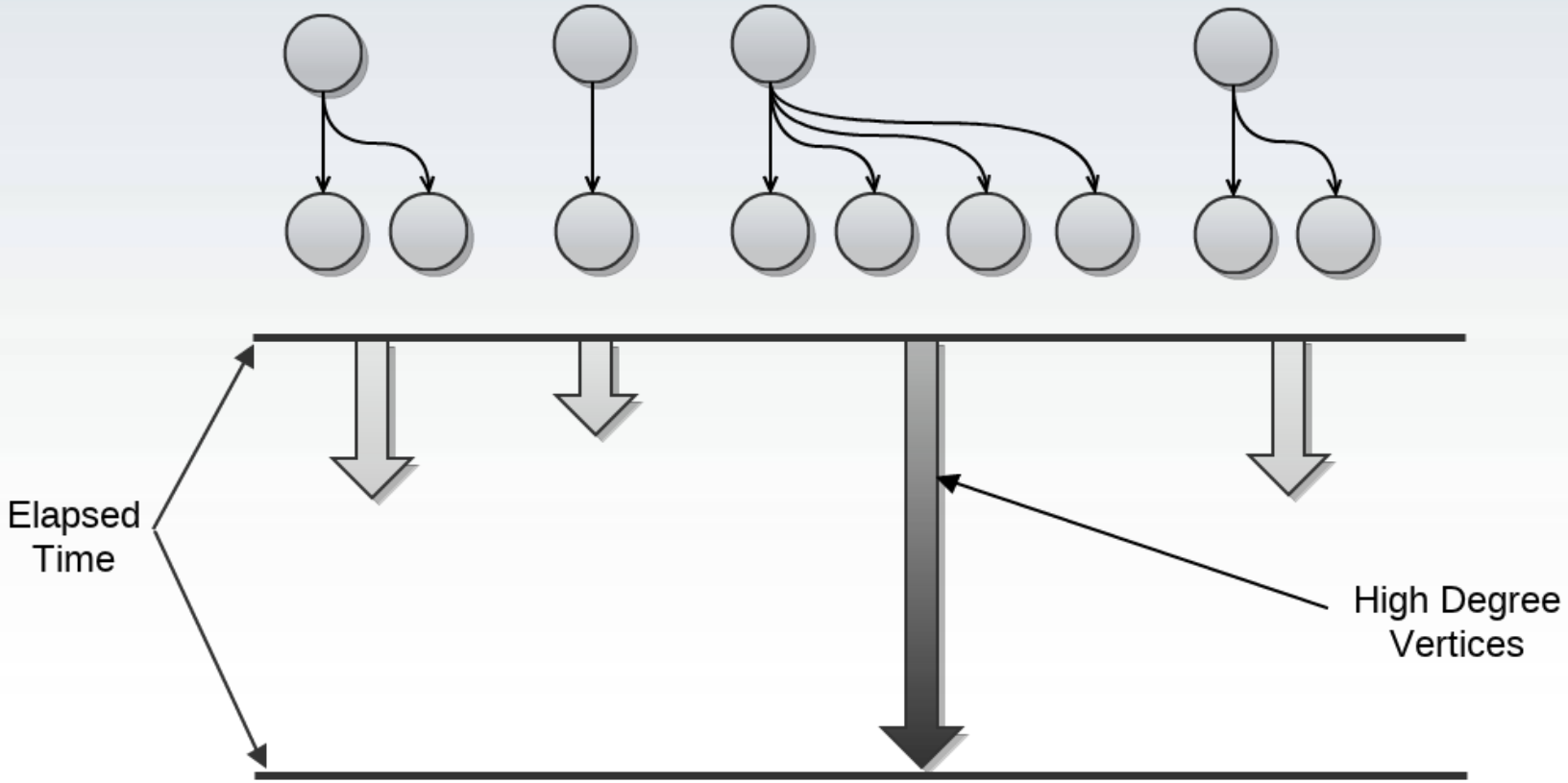
为什么小世界网络最适合并行BFS？





图广度优先搜索算法的并行化方法研究

为什么不规则图最不适合并行BFS？



每一轮的时间取决于度数最高的顶点

工作中使用的第三方库和“轮子”

Description	Library	From
对位图中某一位进行原子操作	syncbitops.h	Linux Kernel
CPU亲和性	sched.h	
对内存进行原子操作	/	GCC
在CPU上对程序进行并行化	omp.h	OpenMP
在GPU上对程序进行并行化	cuda_runtime.h	NVIDIA CUDA
并发队列	concurrent_queue.h	TBB
双向队列	/	STL

THANKS