Parallel and Distributed Optimization with Gurobi Optimizer

Gurobi 优化器的并行和分布计算

GUROBI
OPTIMIZATION

Our Presenter 主讲人

Dr. Greg Glockner
Director of Engineering, Gurobi Optimization, Inc.

Parallel & Distributed Optimization
并行和分布优化

Terminology for this presentation 术语介绍

Parallel computation 并行计算

- One computer 一台机器
  - Multiple processor cores 多线程
  - 1 or more processor sockets 多CPU
- Part of Gurobi throughout our history 与Gurobi 相关
  - MIP branch-and-cut
  - Barrier for LP, QP and SOCP
  - Concurrent optimization

Distributed computation 分布计算

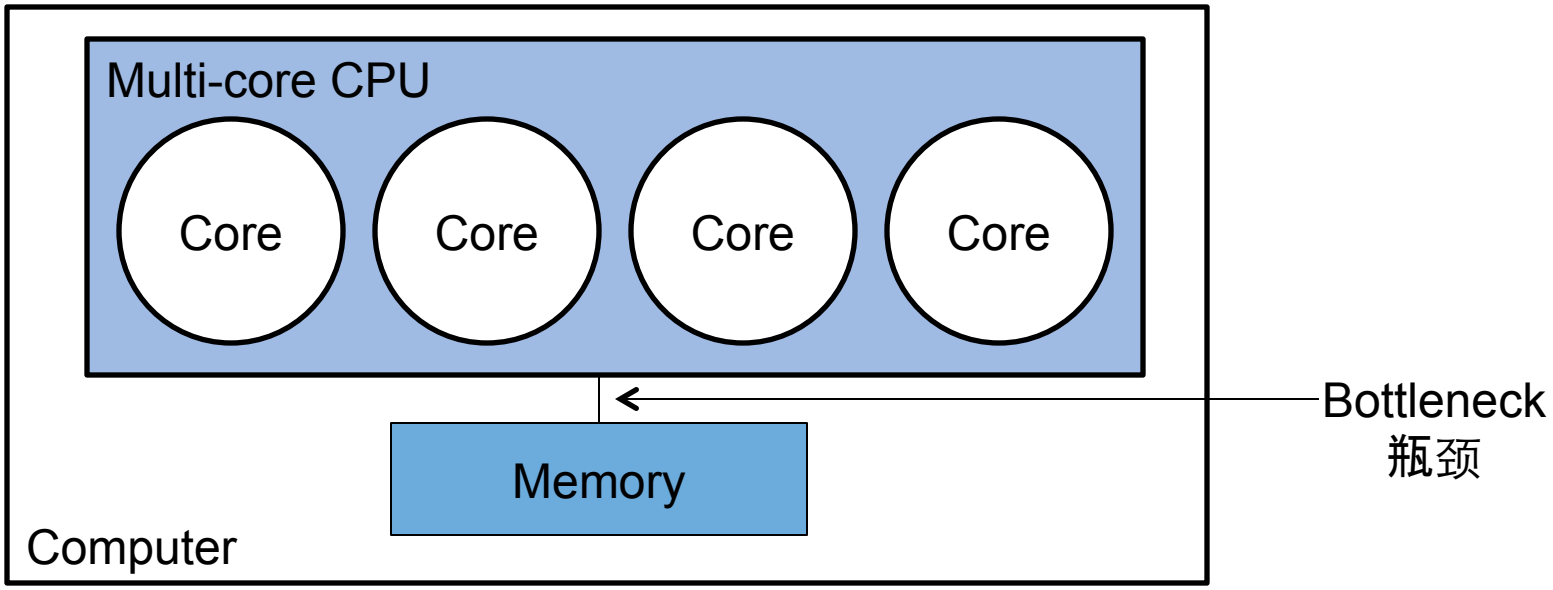
- Multiple computers, linked via a network 局域网内多台互联机器
- Relatively new feature
- Each independent computer can do parallel computation! 每台独立机器可以进行并行计算

Parallel algorithms and hardware 并行算法和硬件

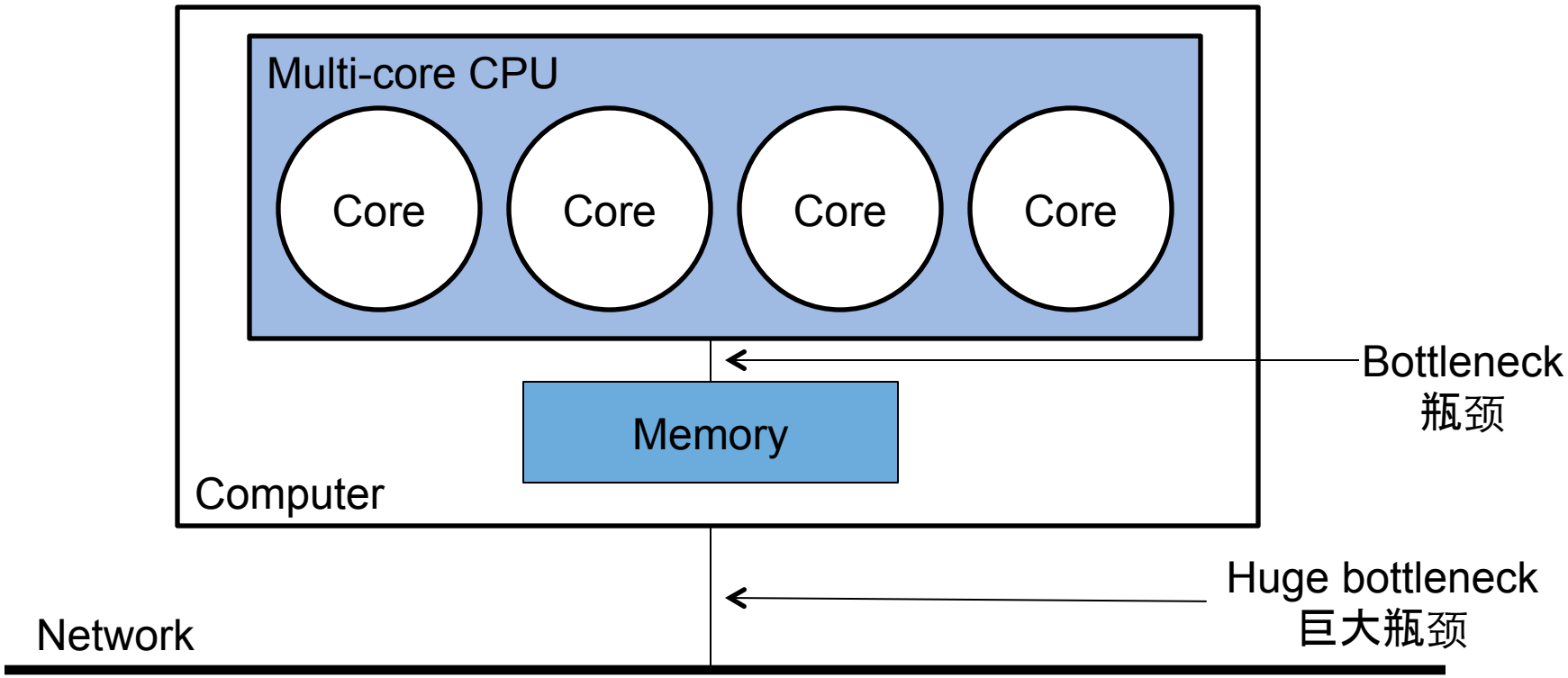
- Parallel algorithms must be designed around hardware 算法围绕硬件设计
  - What work should be done in parallel 并行算法分工
  - How much communication is required 互相通讯
  - How long will communication take 通讯时长

- Goal: Make best use of available processor cores 最大化利用已有硬件条件

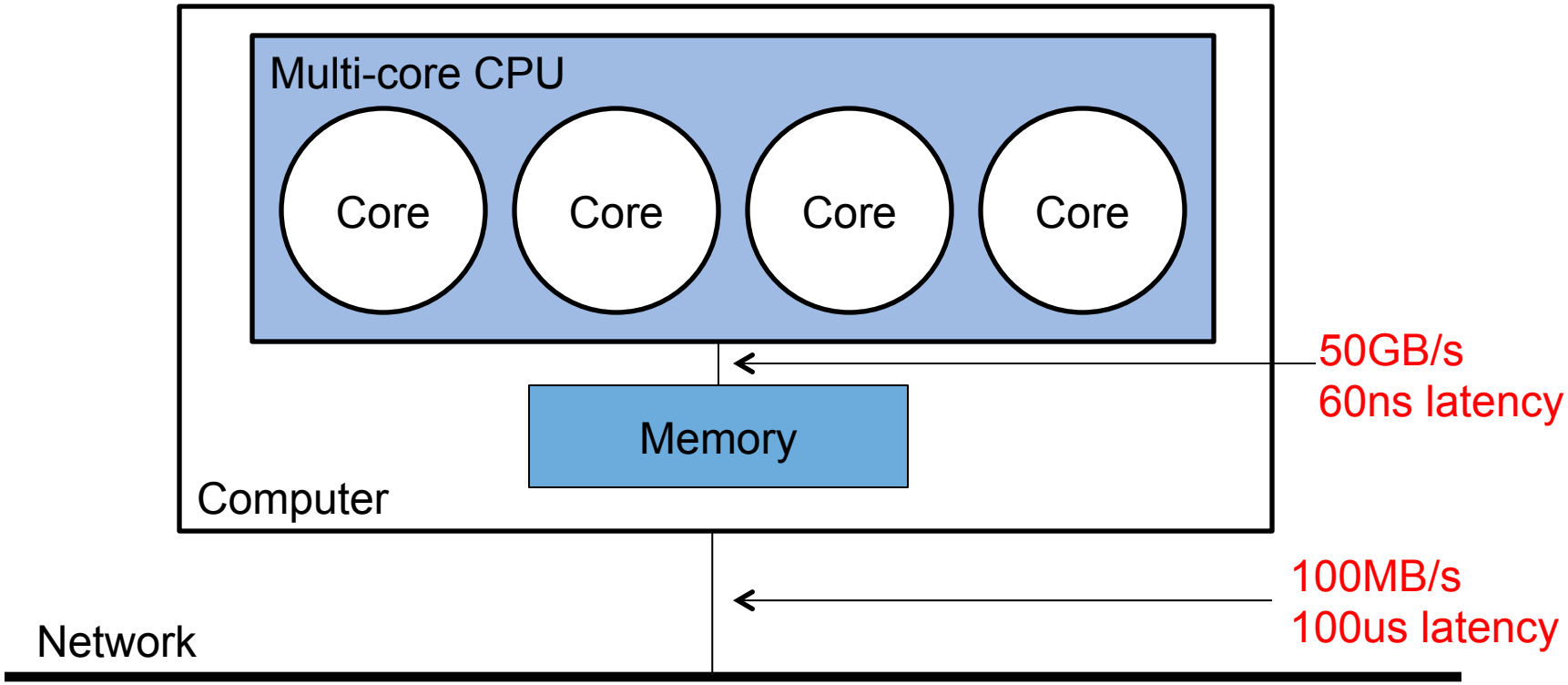
Multi-Core Hardware 多核硬件



Distributed Computing 分布计算



How Slow Is Communication? 通讯有多慢？



- Network is ~1000X slower than memory　网络比内存慢1000倍
  - Faster on a supercomputer, but still relatively slow

Distributed Algorithms in Gurobi 6.0 Gurobi分布算法

- 3 distributed algorithms in version 6.0 三个分布算法
  - Distributed tuning 分布调优
  - Distributed concurrent 分布并发
    - LP (new in 6.0)
    - MIP
  - Distributed MIP (new in 6.0) 分布MIP

Distributed Tuning 分布调优

- Tuning: 调优
  - MIP has lots of parameters
  - Tuning performs test runs to find better settings

- Independent solves are obvious candidate for parallelism
互相独立求解适合并行计算

- Distributed tuning a clear win 分布调优效果明显
  - 10X faster on 10 machines

- Hard to go back once you have tried it

Concurrent Optimization
并发优化

Concurrent Optimization 并发优化

- Run different algorithms/strategies on different machines/cores
  - First one that finishes wins 不同线程运行不同算法，选取先结束者
- Nearly ideal for distributed optimization 基本上适用分布优化
  - Communication:
    - Send model to each machine 将模型发送到各个机器
    - Winner sends solution back 先算完者发回结果
- Concurrent LP: 并发LP
  - Different algorithms: 不同求解方法并发处理
    - Primal simplex/dual simplex/barrier
- Concurrent MIP: 并发 MIP
  - Different strategies 不同策略
  - Default: vary the seed used to break ties 默认不同随机种子
- Easy to customize via concurrent environments 容易客户化

MIPLIB 2010 Testset 测试数据

- MIPLIB 2010 test set…
  - Set of 361 mixed-integer programming models
  - Collected by academic/industrial committee
- MIPLIB 2010 benchmark test set…
  - Subset of the full set – 87 of the 361 models
    - Those that were solvable by 2010 codes
    - (Solvable set now includes 206 of the 361 models)

- Notes: 并非高性能测试数据集
  - Definitely not intended as a high-performance computing test set
    - More than 2/3 solve in less than 100s
    - 8 models solve at the root node
    - ~1/3 solve in fewer than 1000 nodes

Distributed Concurrent MIP 分布式并发 MIP

- Results on MIPLIB benchmark set (>1.00X means concurrent MIP is faster):
  - 4 machines vs 1 machine:

Runtime	Wins	Losses	Speedup
>1s	38	20	1.26X
>100s	17	3	1.50X

  - 16 machines vs 1 machine:

Runtime	Wins	Losses	Speedup
>1s	54	19	1.40X
>100s	26	1	2.00X

Customizing Concurrent 客户化并发策略

- Easy to choose your own settings:
  - Example – 2 concurrent MIP solves:
    - Aggressive cuts on one machine 一台机器采用激进的切平面策略
    - Aggressive heuristics on second machine 另外一台机器采用激进的启发式

    - Java example

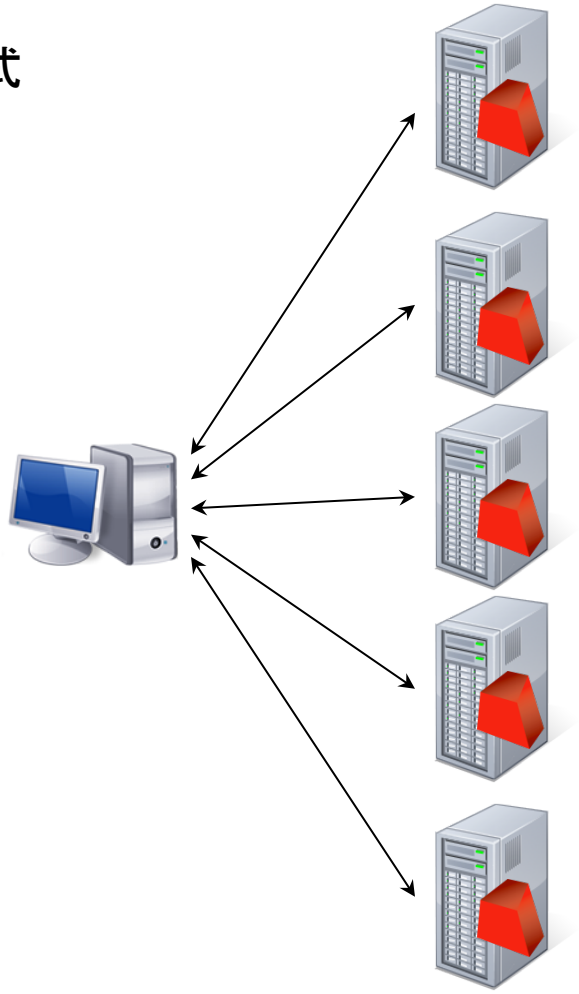
```
GRBEnv env0 = model.getConcurrentEnv(0);
GRBEnv env1 = model.getConcurrentEnv(1);
env0.set(GRB.IntParam.Cuts, 2);
env1.set(GRB.DoubleParam.Heuristics, 0.2);
model.optimize();
model.discardConcurrentEnvs();
```

    - Also supported in C++, .NET, Python and C

Distributed MIP
分布式 MIP

Distributed MIP Architecture 分布式 MIP 架构

- Manager-worker paradigm 管理机-工作机 模式
- Manager 管理机
  - Send model to all workers
  - Track dual bound and worker node counts
  - Rebalance search tree to put useful load on all workers
  - Distribute feasible solutions
- Workers 工作机
  - Solve MIP nodes
  - Report status and feasible solutions
- Synchronized deterministically



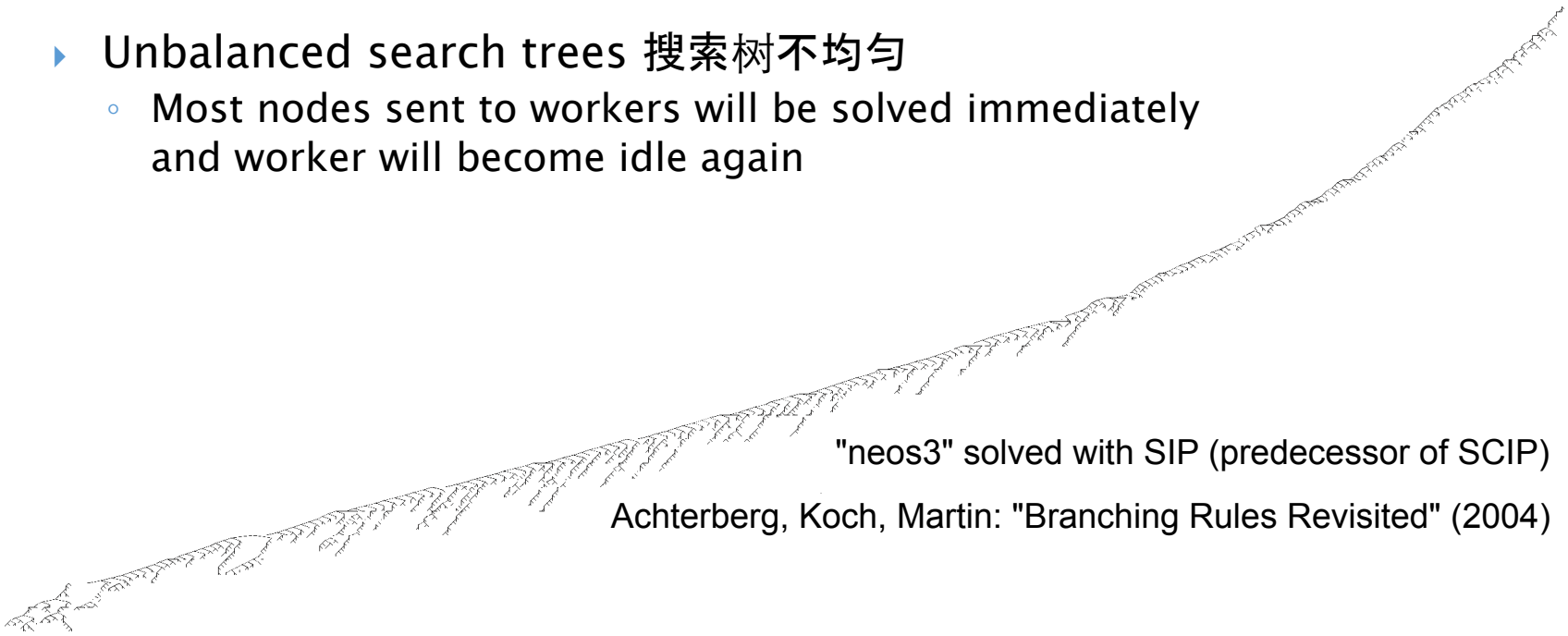
Distributed MIP Phases 分布式 MIP 阶段

- Racing ramp-up phase 预热阶段
  - Distributed concurrent MIP
    - Solve same problem individually on each worker, using different parameter settings
    - Stop when problem is solved or “enough” nodes are explored
    - Choose a “winner” – worker that made the most progress

- Main phase 主算阶段
  - Discard all worker trees except the winner's
  - Collect active nodes from winner, distribute them among now idle workers
  - Periodically synchronize to rebalance load

Bad Cases for Distributed MIP 分布式 MIP 不适用于

- Easy problems 简单问题
  - Why bother with heavy machinery?
- Small search trees 搜索树太小
  - Nothing to gain from parallelism
- Unbalanced search trees 搜索树不均匀
  - Most nodes sent to workers will be solved immediately and worker will become idle again

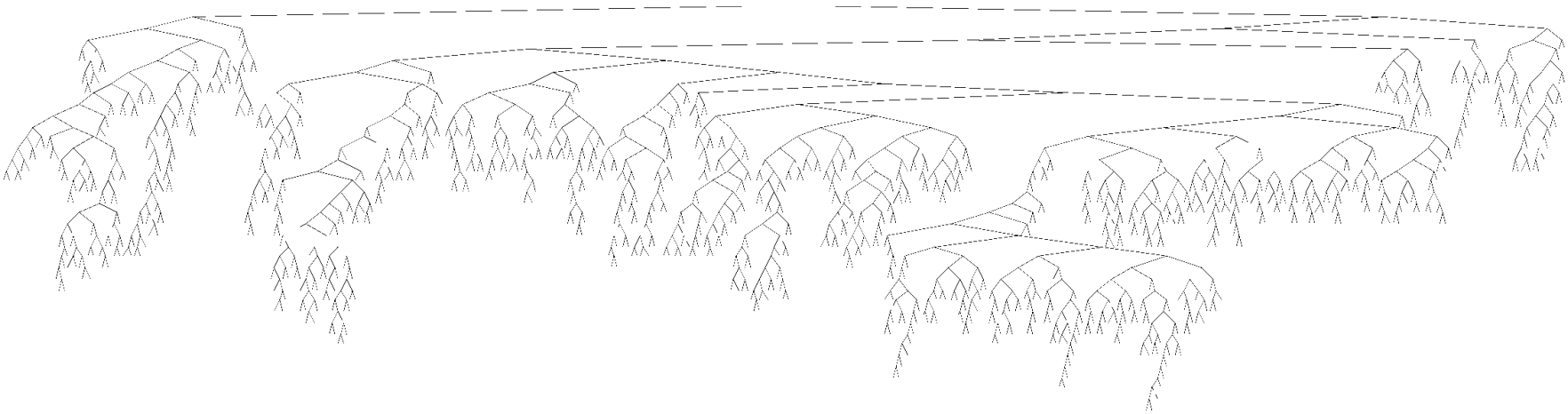


"neos3" solved with SIP (predecessor of SCIP)

Achterberg, Koch, Martin: "Branching Rules Revisited" (2004)

Good Cases for Distributed MIP 分布式 MIP 适合

- Large search trees 庞大搜索树
- Well-balanced search trees 分布均匀的搜索树
  - Many nodes in frontier lead to large sub-trees



"vpm2" solved with SIP (predecessor of SCIP)

Achterberg, Koch, Martin: "Branching Rules Revisited" (2004)

Performance
性能

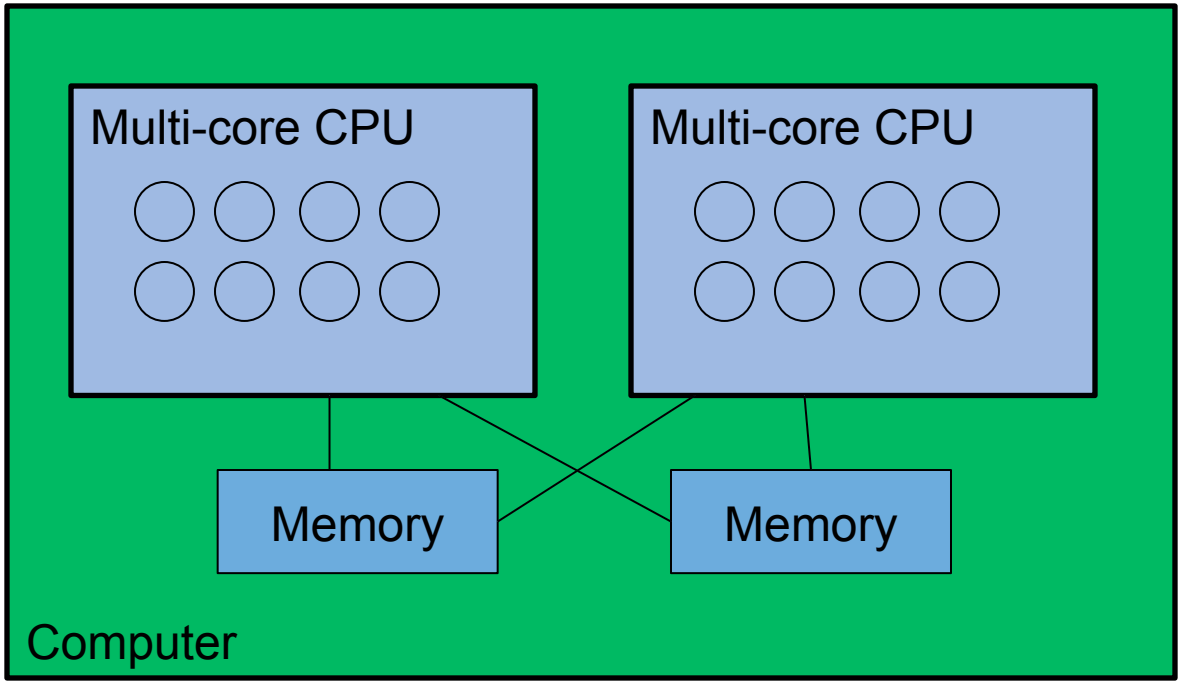
Three Views of 16 Cores　三种利用16核的方案

- Consider three different tests, all using 16 cores:
  - On a 16-core machine:
    - Run the standard parallel code on all 16 cores　一台16核运行并发代码
    - Run the distributed code on four 4-core subsets　分成4个4核，运行分布式代码
  - On four 4-way machines:
    - Run the distributed code　分成4台机器，每台4核，运行分布式代码

- Which gives the best results? 哪个结果最好？

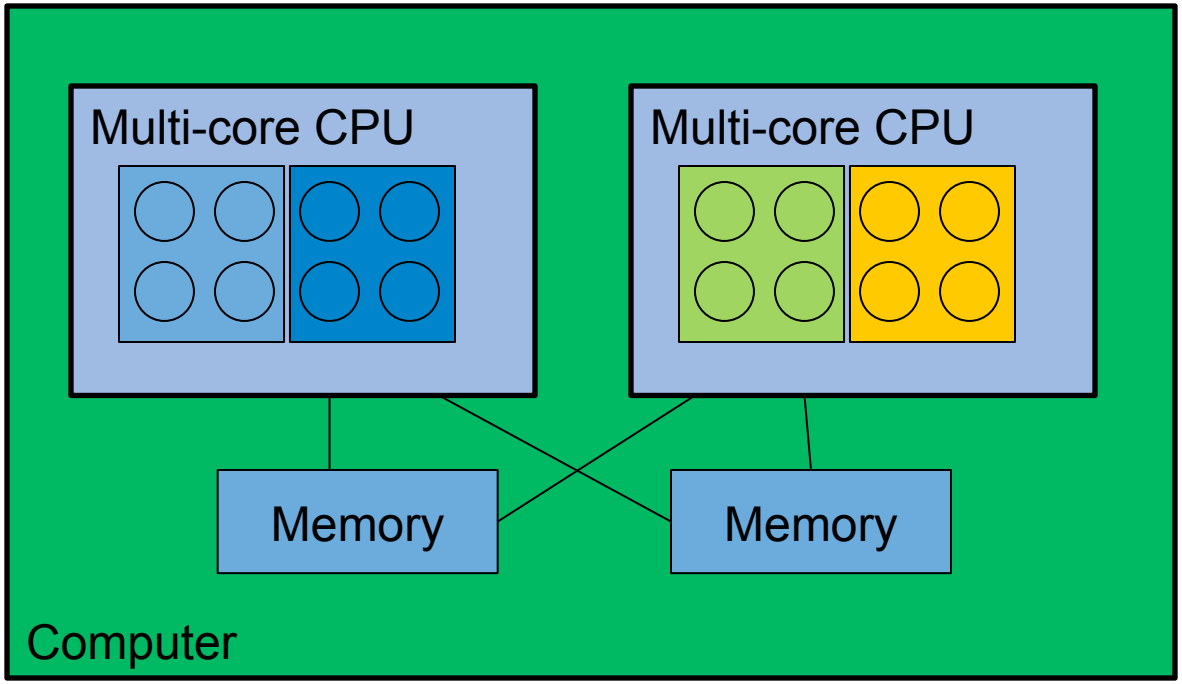
Parallel MIP on 1 Machine　一台机器16核并行

- Use one 16-core machine:



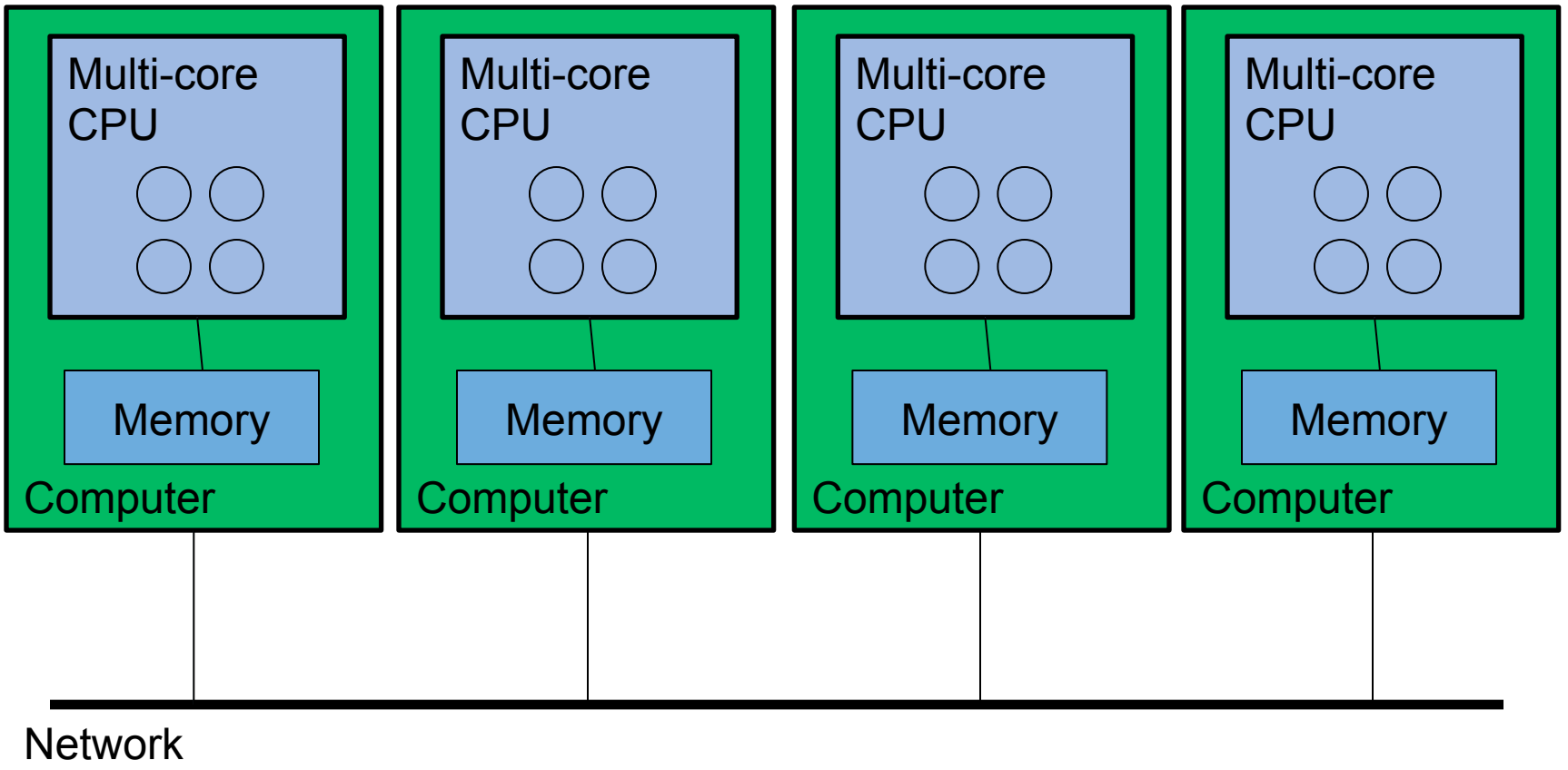
Distributed MIP on 1 machine 一台机器4*4 分布式

- Treat one 16-core machine as four 4-core machines:



Distributed MIP on 4 machines 4台机器 分布式

- Use four 4-core machines



Performance Results 性能结果

- Using one 16-core machine (MIPLIB 2010, baseline is 4-core run on the same machine)…

Config	>1s	>100s
One 16-core	1.57X	2.00X
Four 4-core	1.26X	1.82X

- Better to run one-machine algorithm on 16 cores than treat the machine as four 4-core machines 将一台机器16核集中使用 比 4*4 分布式好
  - Degradation isn't large, though

Performance Results 性能结果

- Comparing one 16-core machine against four 4-core machines (MIPLIB 2010, baseline is single-machine, 4-core run)…

Config	>1s	>100s
One 16-core machine	1.57X	2.00X
Four 4-core machines	1.43X	2.09X

- Given a choice… 权衡
  - Comparable mean speedups　加速性能相近
  - Other factors… 其他因素
    - Cost: four 4-core machines are much cheaper　成本
    - Admin: more work to admin 4 machines　管理复杂度

Distributed Algorithms in 6.0　分布式算法

- MIPLIB 2010 benchmark set
  - Intel Xeon E3-1240v3 (4-core) CPU
  - Compare against 'standard' code on 1 machine

Machines	>1s			>100s		
	Wins	Losses	Speedup	Wins	Losses	Speedup
2	40	16	1.14X	20	7	1.27X
4	50	17	1.43X	25	2	2.09X
8	53	19	1.53X	25	2	2.87X
16	52	25	1.58X	25	3	3.15X

Some Big Wins　一些显著提升

- Model *seymour*
  - Hard set covering model from MIPLIB 2010
  - 4944 constraints, 1372 (binary) variables, 33K non-zeroes

Machines	Nodes	Time (s)	Speedup
1	476,642	9,267s	-
16	1,314,062	1,015s	9.1X
32	1,321,048	633s	14.6X

Distributed Concurrent Versus Distributed MIP
分布式并发 对比 分布式 MIP

- MIPLIB 2010 benchmark set (versus 1 machine run):
  - >1s

Machines	Concurrent	Distributed
4	1.26X	1.43X
16	1.40X	1.58X

  - >100s

Machines	Concurrent	Distributed
4	1.50X	2.09X
16	2.00X	3.15X

Gurobi Distributed MIP 分布式 MIP

- Makes huge improvements in performance possible 显著提升变得可能
- Mean performance improvements are significant but not huge 性能提升显著，但不巨大
  - Some models get big speedups, but many get none 一些模型成效显著
  - Much better than distributed concurrent 比分布式并发更好
  - As effective as adding more cores to one box 相当于为一个机器增加核数
- Effectively exploiting parallelism remains: 并行计算仍然
  - A difficult problem 困难重重
  - A focus at Gurobi Gurobi 关注焦点

Mechanics
操作方法

Gurobi Remote Services Gurobi 远程服务

- Install Gurobi Remote Services on worker machines 在工作机上安装远程服务
  - No Gurobi license required on workers 工作机上无需许可
  - Machine listens for Distributed Worker requests 监测请求
- Set a few parameters on manager 在管理机上设定参数
  - `ConcurrentJobs=4`
  - `WorkerPool=machine1,machine2,machine3,machine4`
  - No other code changes required
- Manager must be licensed to use distributed algorithms 管理机需要分布式算法许可
  - Gurobi Distributed Add-On
    - Enables up to 100 workers 支持最多100台工作机

Integral Part of Product 产品的组成部分

- Built on top of Gurobi Compute Server 基于Gurobi 计算服务器功能
  - Only 1500 lines of C code specific to concurrent/distributed MIP
- Built into the product 无需额外安装
  - No special binaries involved

- Bottom line: 总结, 以常规 MIP算法为基础, 同步提升
  - Changes to MIP solver automatically apply to distributed code too
    - Performance gains in regular MIP also benefit distributed MIP
  - Distributed MIP will evolve with regular MIP

Footnote: GPGPU computing GPU 计算

- GPGPU: General-purpose computing on Graphics Processing Units
  - Massively parallel for simple computation 并行简单计算
  - Heavily marketed for parallel tasks 并行任务处理

- Currently, GPUs are not well-suited for solvers like Gurobi GPU 不适合 Gurobi 及其他优化器
  - For LP, sparse linear algebra does not parallelize to hundreds of GPUs
  - For MIP, each tree node requires very different calculations, but GPU SIMD computations are designed for identical calculations on different data

- General-purpose CPUs continue to add parallel cores, which benefit Gurobi Optimizer 通用 CPU 增加核数，有益于 Gurobi

Licensing 许可

- Commercial 商用
  - Not included – must purchase the distributed option
  - Ask your sales representative for benchmarks or pricing

- Academic 学术
  - Named-user: not included in licenses from Gurobi website
  - Site license: not currently supported
  - If interested, your network administrator must contact Gurobi support to request a single-machine, distributed license