基于应用运行特征的快速优化方法

如何提高应用程序运行速度
--从计算机软件优化角度

Intel软件服务商：北京并行科技有限公司

http://www.paratera.com

几个常用的应用性能指标

- Performance (Wall time)
- CPU Utilization (CPU利用率<100%，一定有系统瓶颈)
- Frequency Scaling (性能随CPU主频如何变化)
- Core Scaling (性能随CPU核数的增加如何变化)
- Cluster Scalability(机群节点加速比)

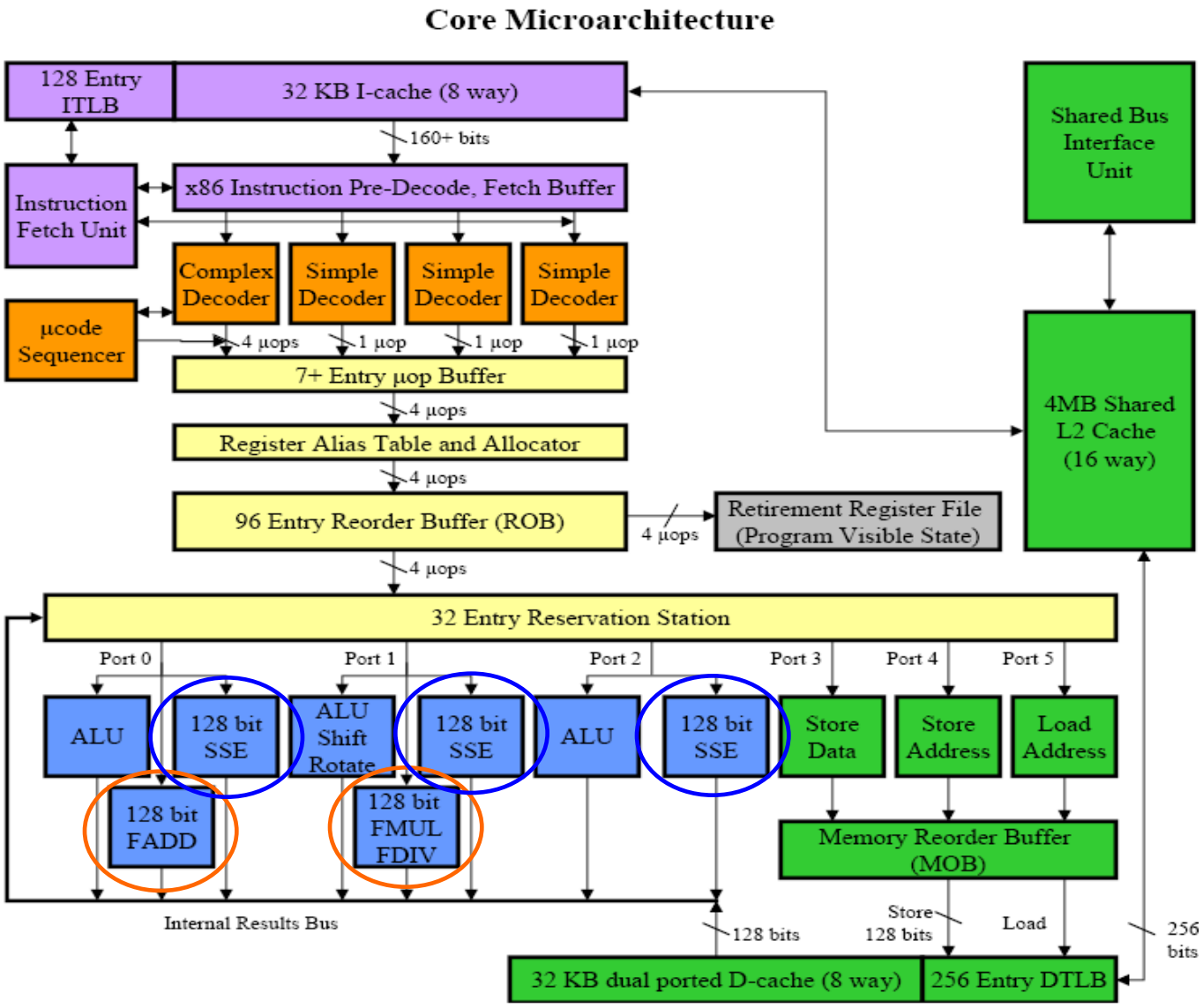
性能的定义

- 什么是性能？
- Performance(Time/Bandwidth/latency/...)= F(Hardware+Software+Workload+Benchmark)
- 性能（时间/带宽/延迟/...）= F(特定硬件+特定软件+特定负载+特定测量方法)
- 硬件性能
  - 标准Benchmark在硬件上的运行结果
  - Linpack/Stream/Iozone/IMB/...
- 软件性能
  - 特定软件在某个硬件平台上的运行结果
  - Fluent/Ansys/MM5/Gaussian/GeoEast/...

软件是否释放硬件潜能

- 现代计算机体系结构设计
  - 超标量
  - 乱序执行
  - 向量化
  - Cache层次结构
  - 多核
  - 多处理器
  - FSB/NUMA
  - 高速互联网络
  - 并行文件系统
  - ……
- 基于现代硬件特征，如何做性能优化？

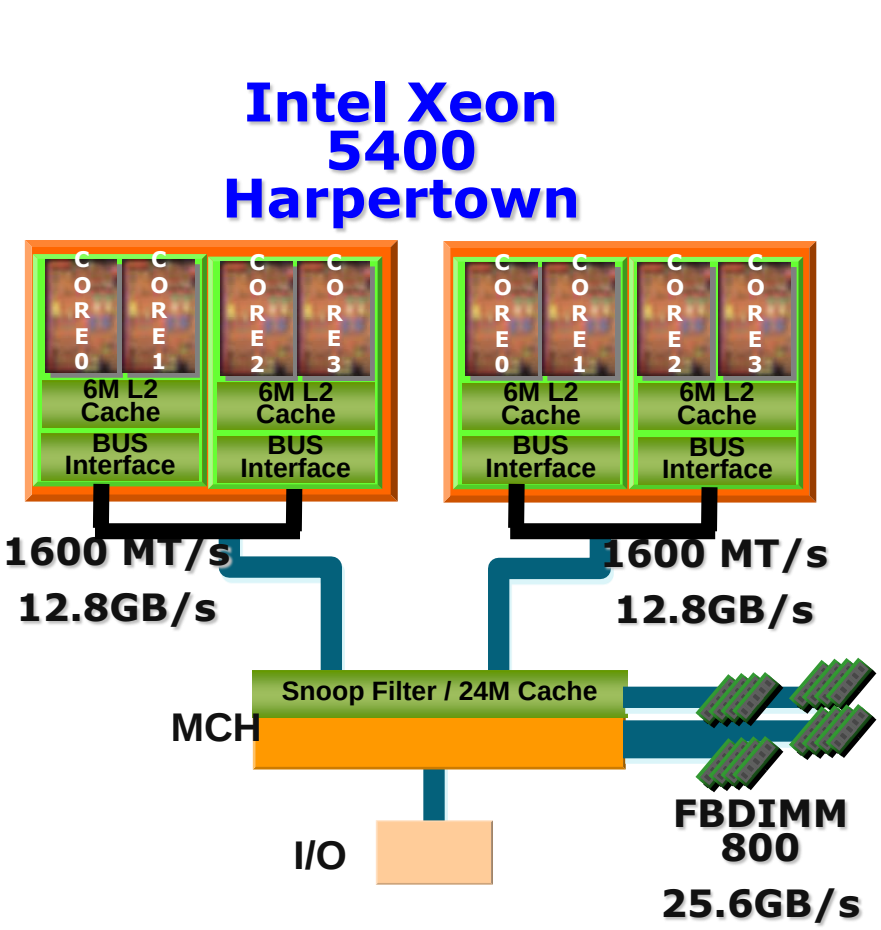
Intel Core Microarchitecture

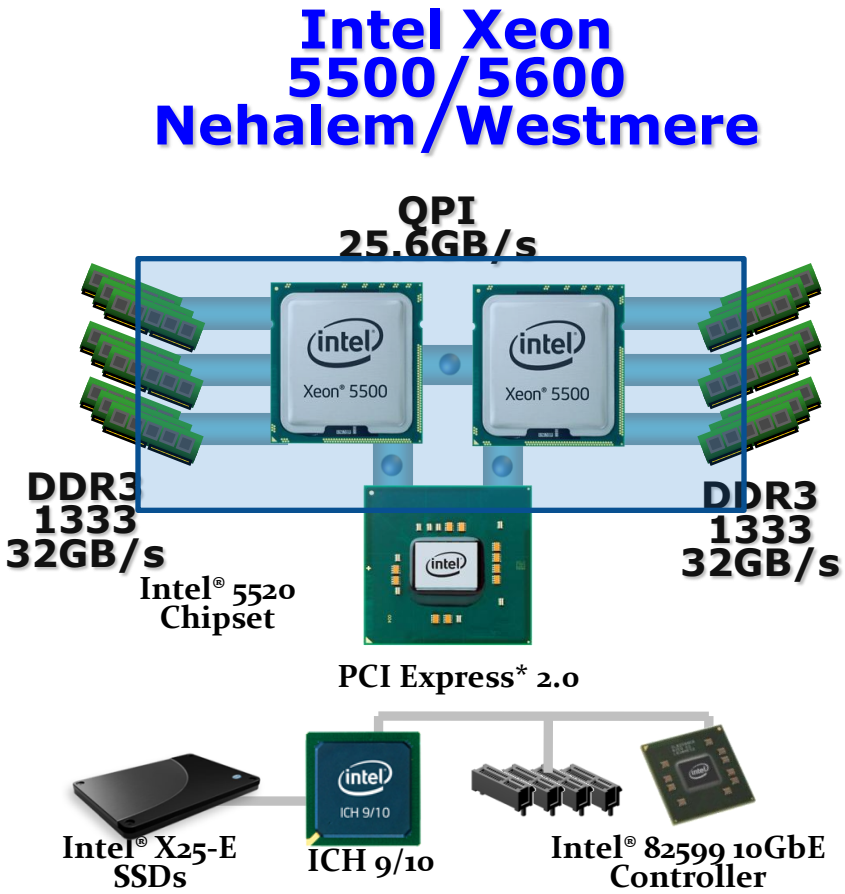


SIMD处理单元

浮点处理单元

Intel Xeon Platform



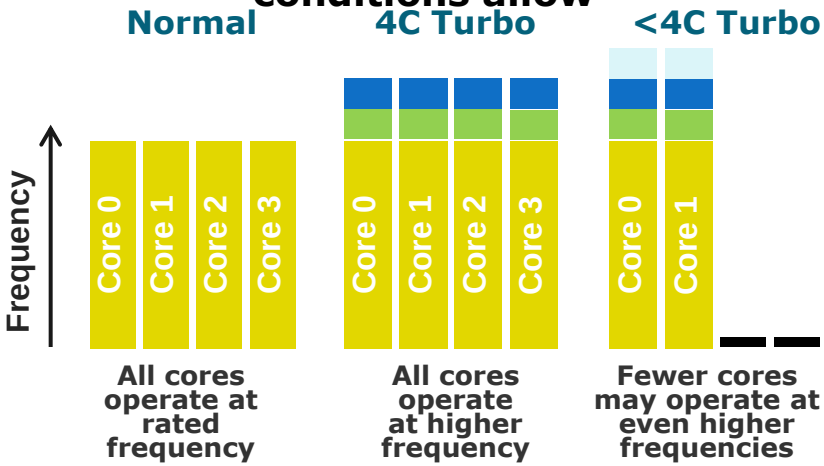


Hero Features

Intel® Xeon® 5500/5600 Series Processor (Nehalem/Westmere)

Intel® Turbo Boost Technology

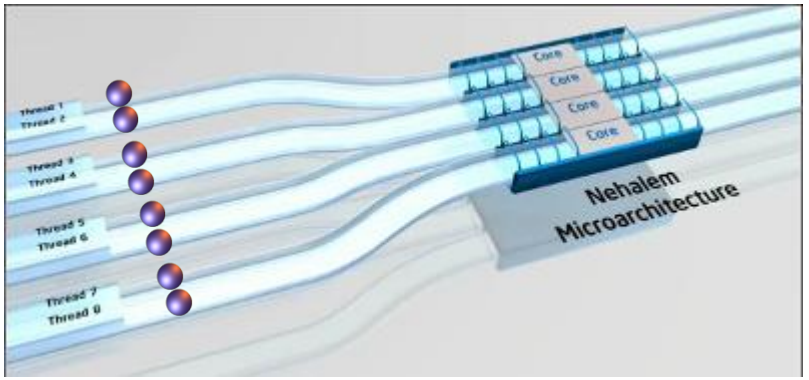
Increases performance by increasing processor frequency and enabling faster speeds when conditions allow



Higher performance on demand

Intel® Hyper-Threading Technology

Increases performance for threaded applications delivering greater throughput and responsiveness



Higher performance for threaded workloads

高性能计算当前问题

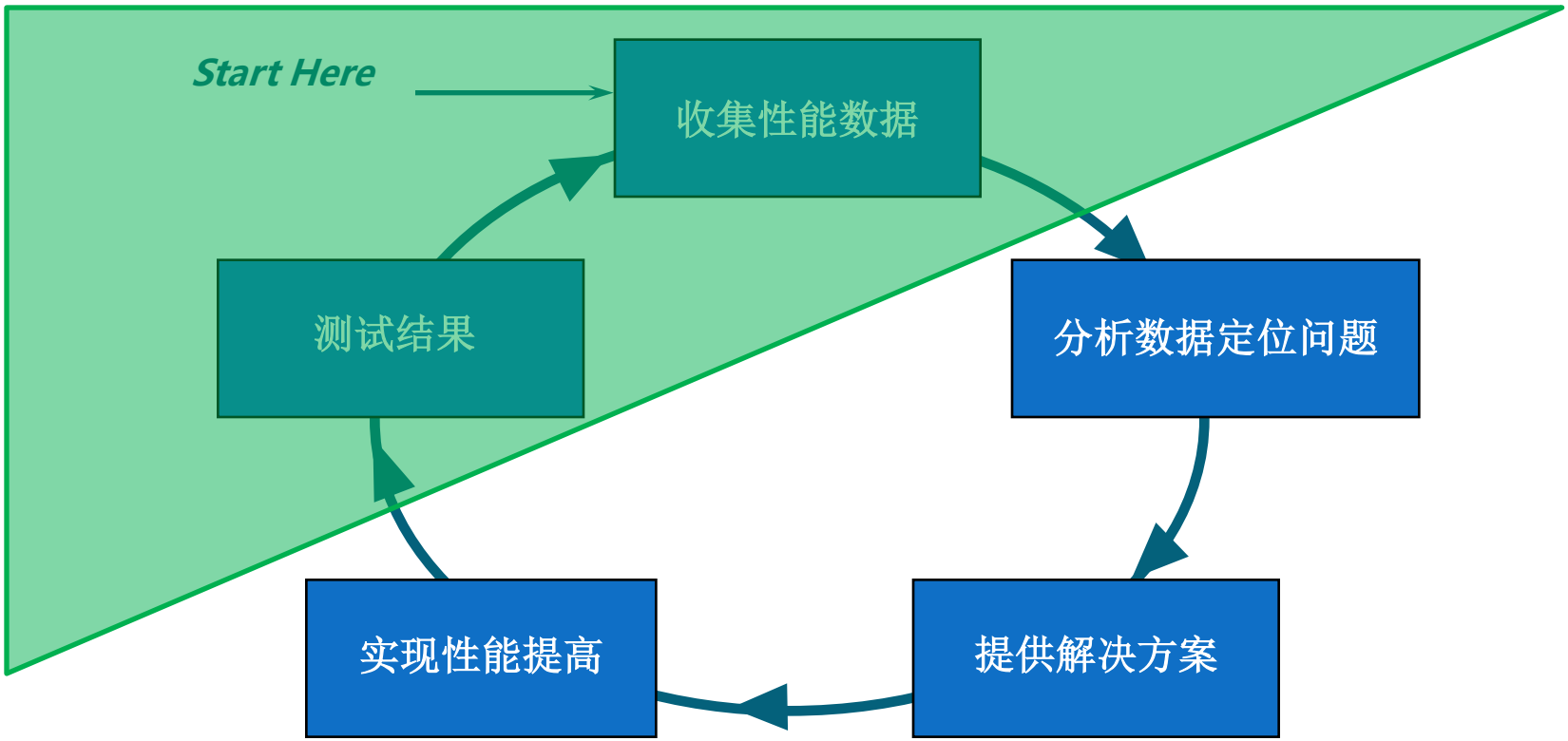
- 绝大多数实际运行的软件只发挥硬件很少的计算能力
- 软件优化技术太复杂，需要专业工具和专家支持
- 软件优化成本太高，耗时太长，需要快速优化方法

目标：用20%的时间挖掘出80%的优化空间

解决方案：基于应用运行特征的快速优化方法

性能数据可视化：性能看得见

性能优化循环

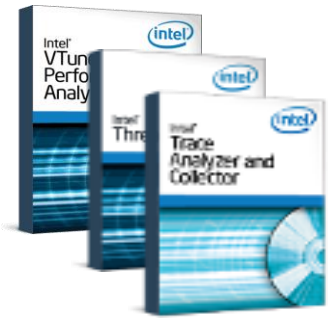


节约测试和收集数据时间，有效时间用于分析和优化

应用运行特征分析方法

- 确定合适的测试应用和负载
- 收集CPU/Memory/Disk/Netowrk四部分的主要特征指标
- 找到影响应用性能的最重要的几个技术指标，设计相应的性能评估模型（变动某个指标会对性能造成什么样的影响）
- 综合考虑动态/暂态特征和稳态特征
- 提供数据支持，辅助用户综合考虑 性能/功率/价格 设计出优化机群系统方案
- 使用性能剖析工具 Paramon/Intel软件实现自动，低干扰应用运行特征采集

软件性能示波器 = **软件** + **专家**

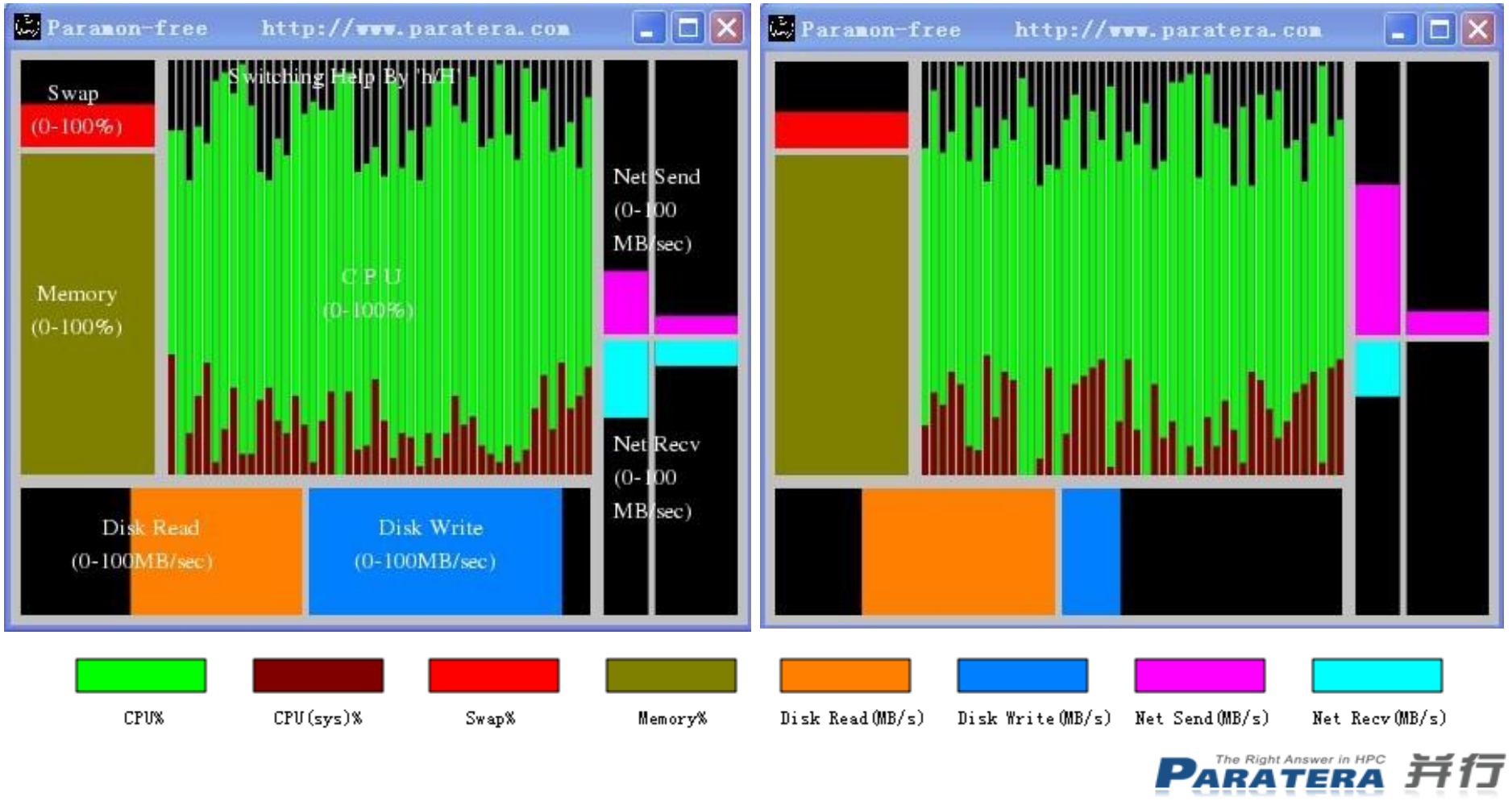


主要性能特征指标

- 系统级
  - CPU: 应用计算时间、系统开销时间
  - Memory: 内存使用率、SWAP使用率
  - Disk：磁盘读、磁盘写
  - Network：网络收、网络发
- 微架构级
  - 代码效率：CPI、GIPS
  - 浮点指令：x87、SSE、sSSE
  - 向量化：fVEC
  - 内存访问：Memory Bandwidth

Paramon性能监控

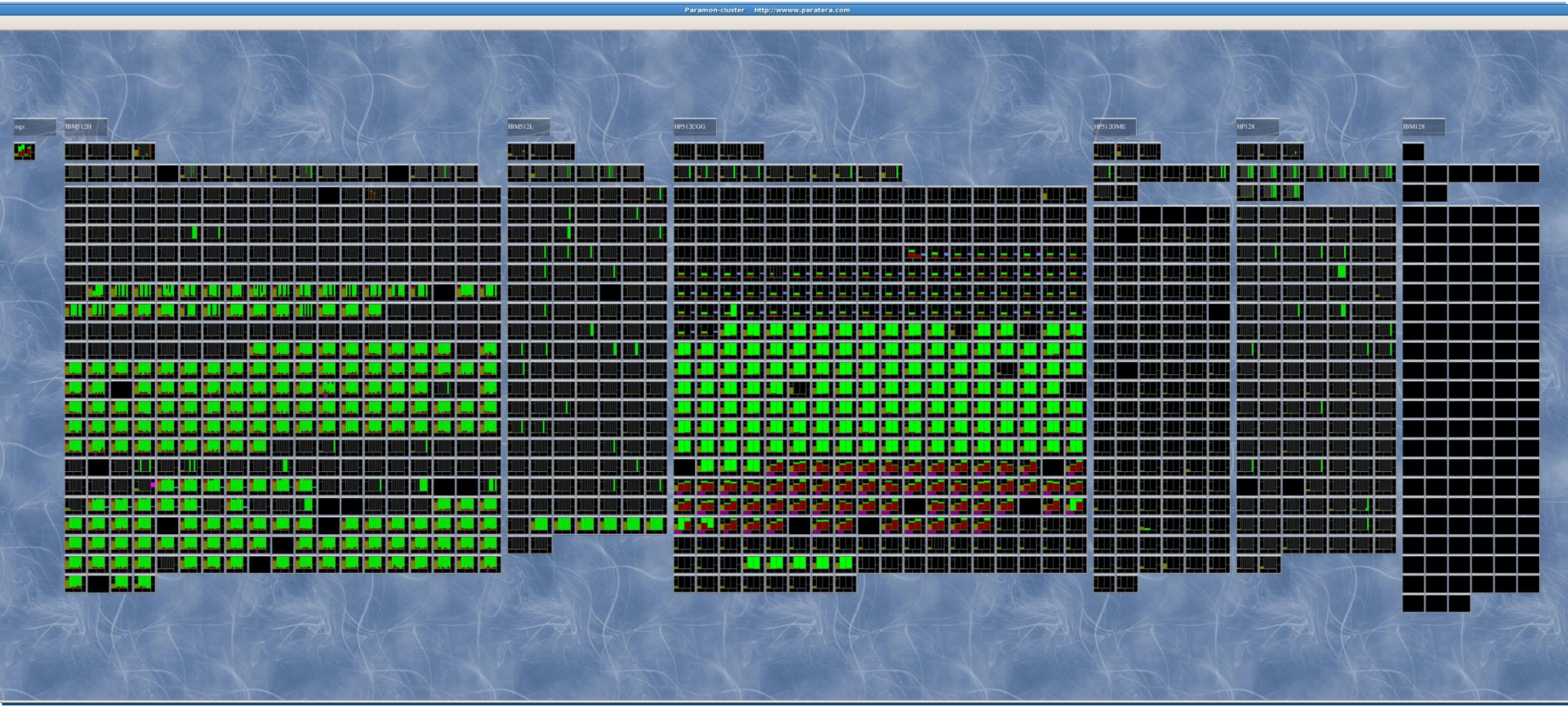
Paramon-free单机版的主界面(http://www.paratera.com直接下载)



Paramon性能监控(406结点)

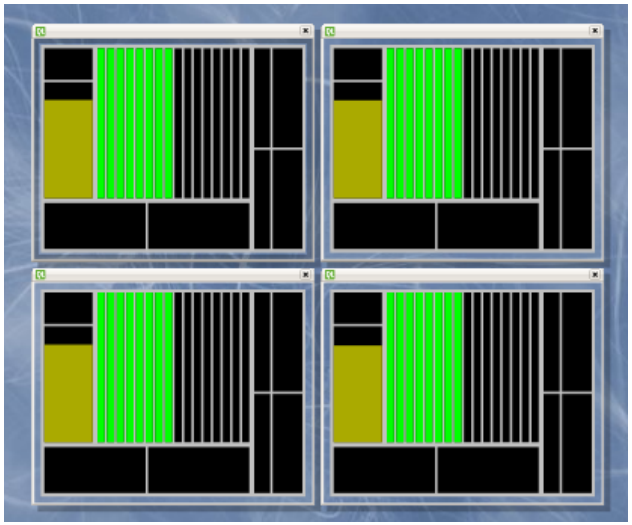
Paramon使用案例

- 某油田1350节点六套机群

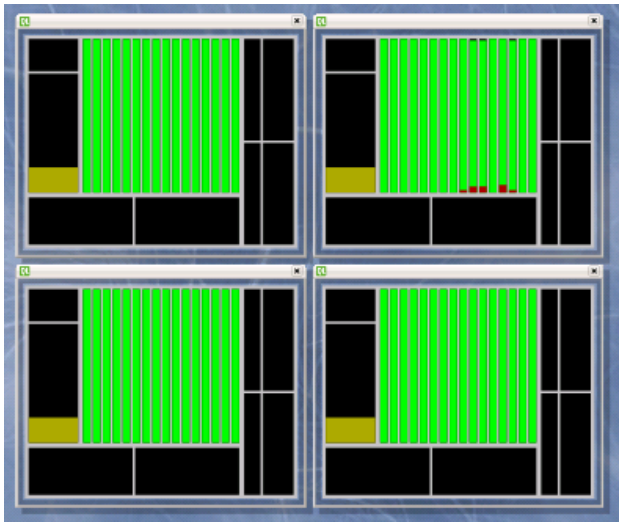


Paramon使用案例

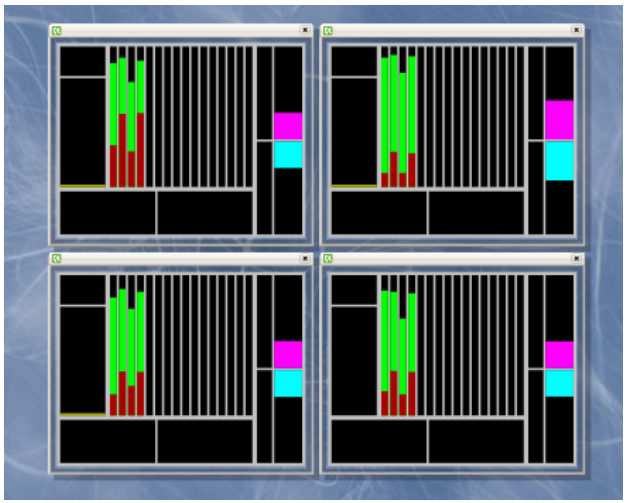
- 并行科技Para+（4节点）基准测试(CPU/Memory/Network/Disk)



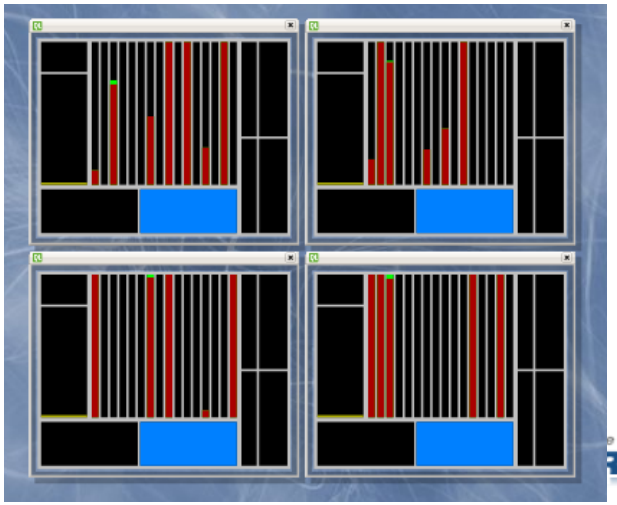
Linpack测试



Stream测试



IMB测试



IOzone测试

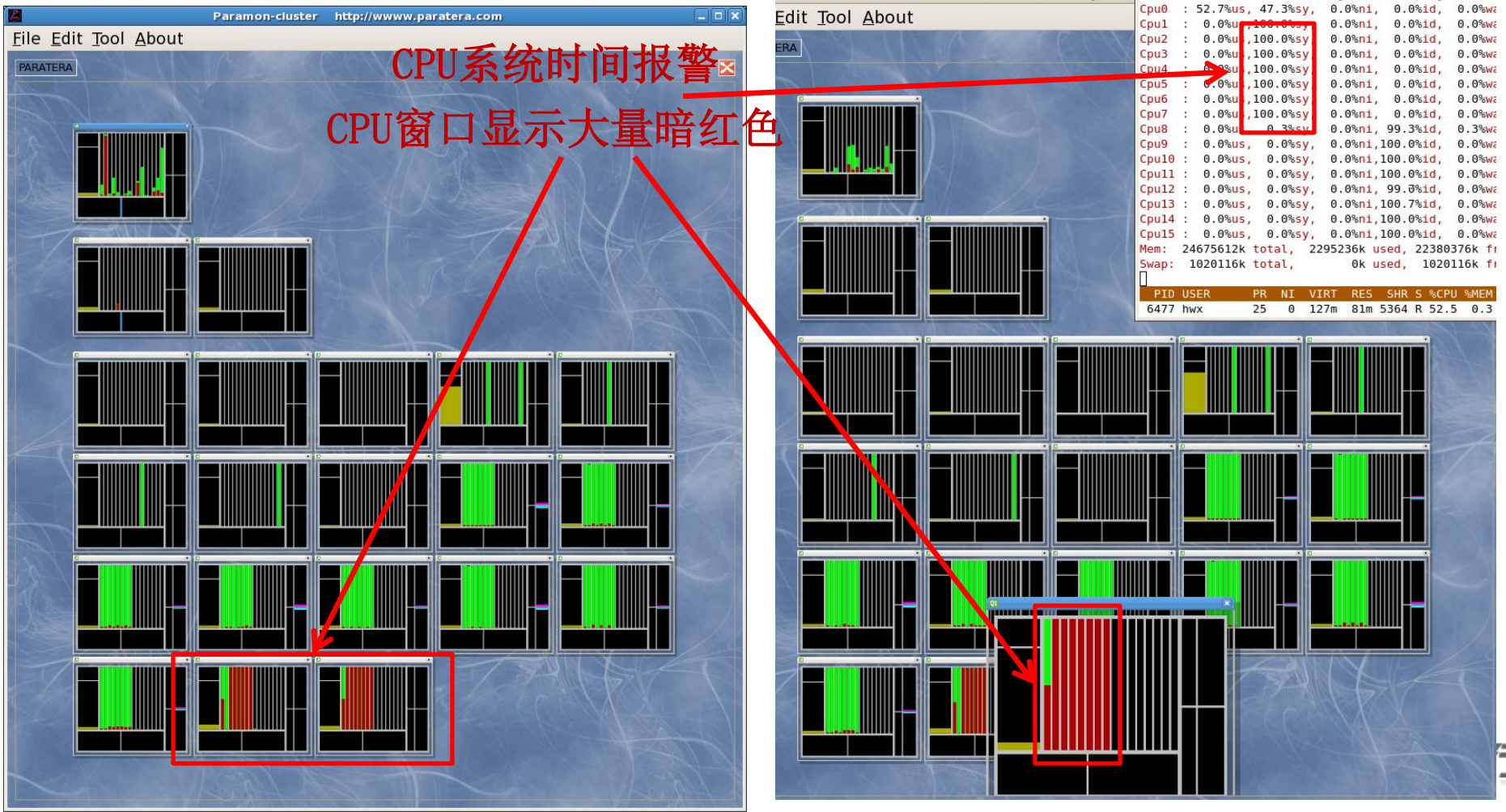
Paramon使用案例

- 并行科技Para+（4节点）基准测试(CPU/Memory/Network/Disk)

序号	基准测试名称	测试内容
1	Linpack测试	测试节点间的CPU计算性能
2	Stream测试	测试节点间Memory内存带宽
3	Intel MPI Benchmark测试（简称IMB测试）	测试各节点间的Network网络性能
4	IOzone测试	测试节点的Disk磁盘读写性能

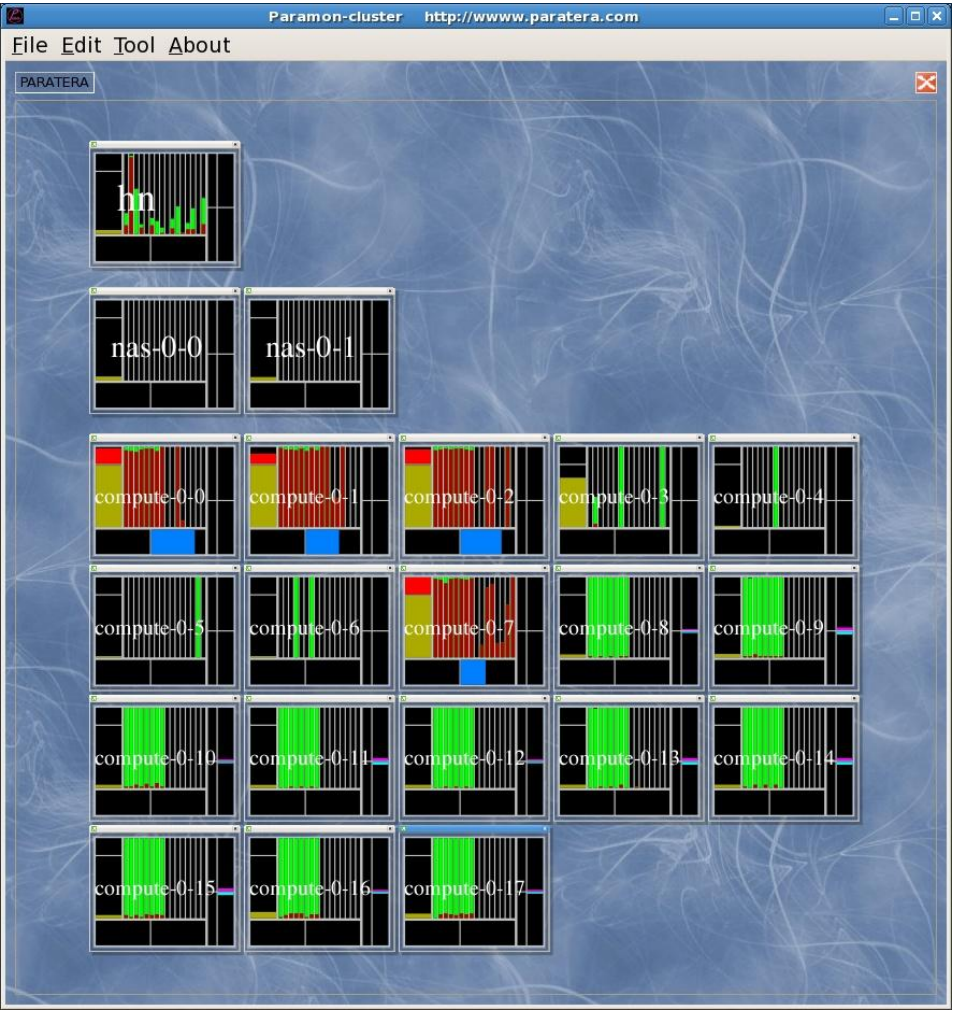
Paramon使用案例

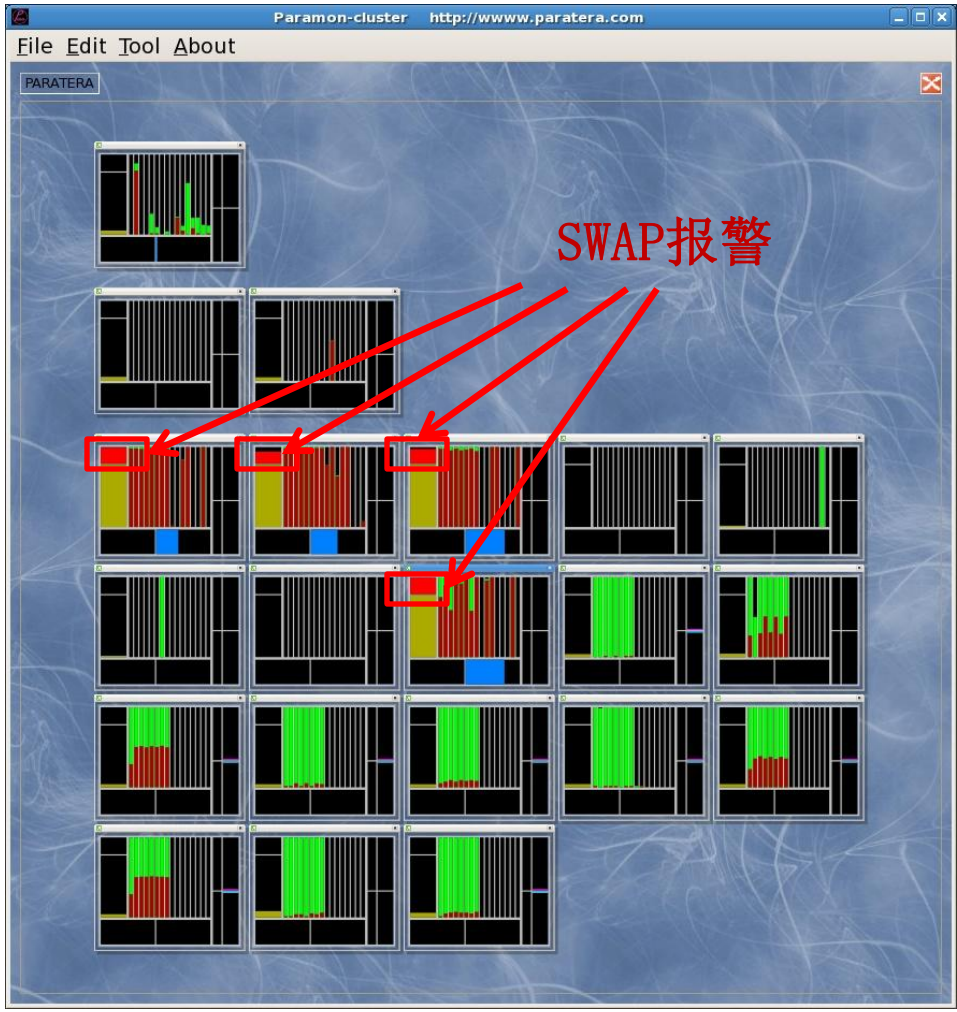
- Paramon报警案例——CPU系统时间报警



Paramon使用案例

- Paramon报警案例——SWAP报警



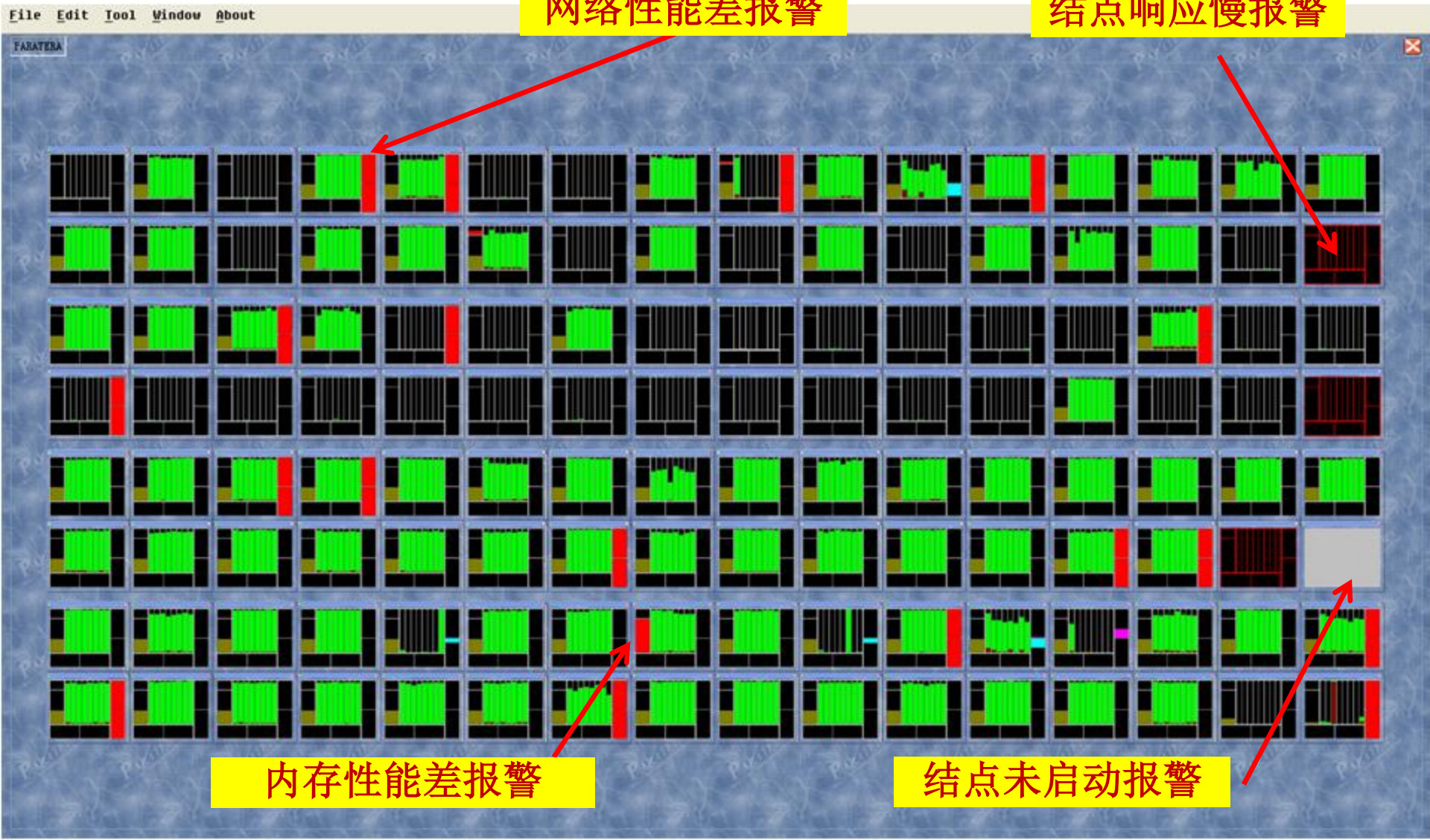


Paramon使用案例

- 除以上报警方式外，Paramon软件还支持以下形式的报警：
  - CPU核数缺失的报警；
  - 内存总量缺失的报警；
  - 磁盘总量使用情况的报警，如：磁盘剩余容量少时；
  - 网络状态的报警，如：千兆位以太网被识别为百兆位以太网；Infiniband 40Gb/s被识别成20Gb/s或更低，导致性能变差等。
- 当发生以上各类问题时，单体窗口界面内的对应部件会相应显示红色。
- 支持用户自定义形式的报警。

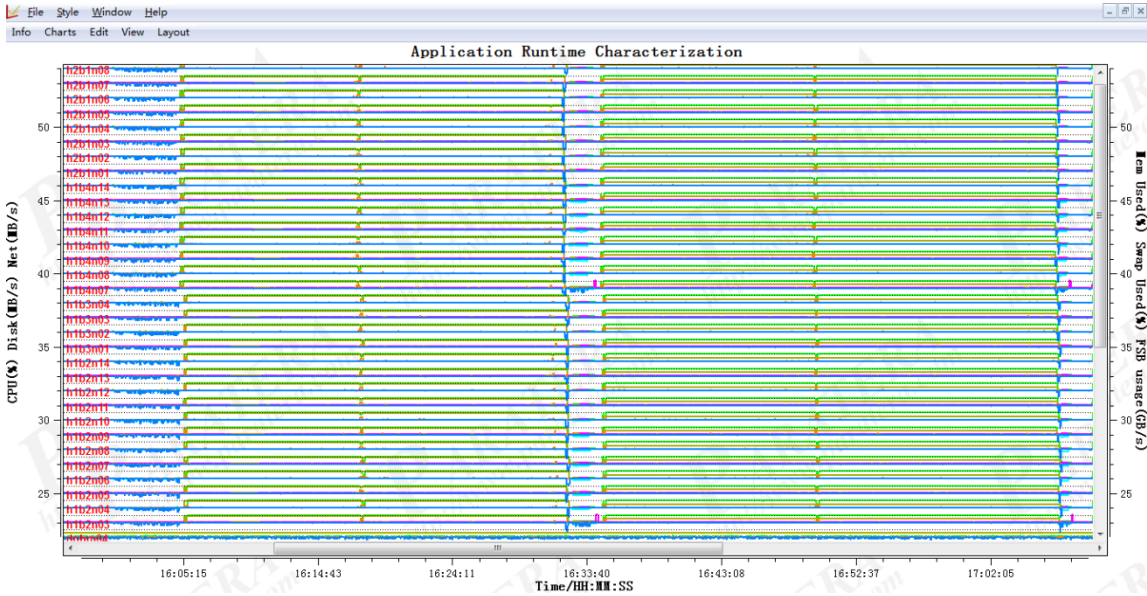
性能看得见

某油田128结点系统性能扫描



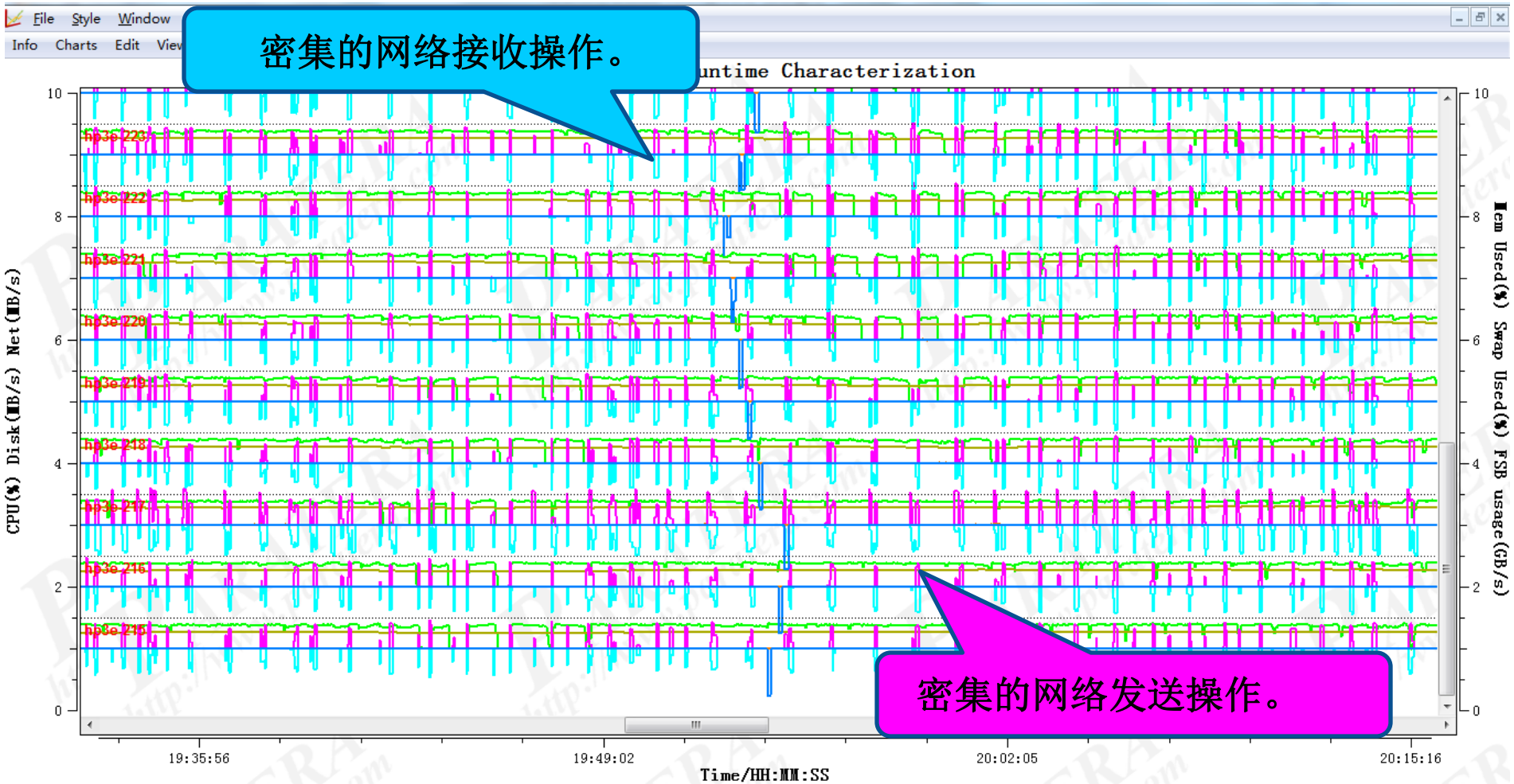
Paratune性能分析

序号	性能指标项	颜色说明	颜色图例
1	CPU使用率	绿色	CPU%
2	CPU SYS使用率	深红色	CPU(sys)%
3	内存使用率	深黄色	Memory%
4	SWAP使用率	红色	Swap%
5	网络发送量	洋红	Net Send
6	网络接收量	青色	Net Recv
7	磁盘读量	橙色	Disk Read
8	磁盘写量	湖蓝色	Disk Write



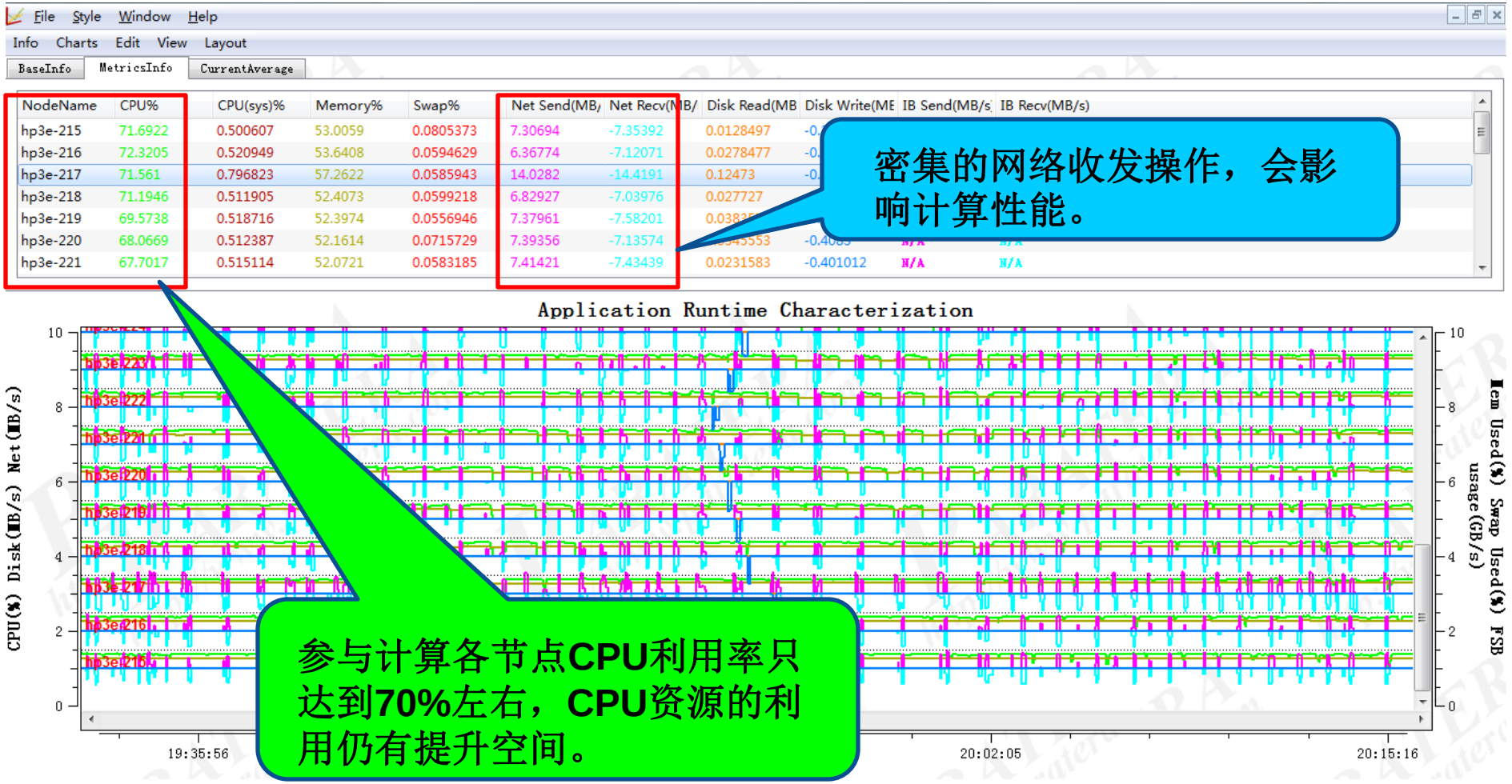
Paratune使用案例

- 案例1——叠前深度偏移模块



Paratune使用案例

- 案例1——叠前深度偏移模块



Paratune使用案例

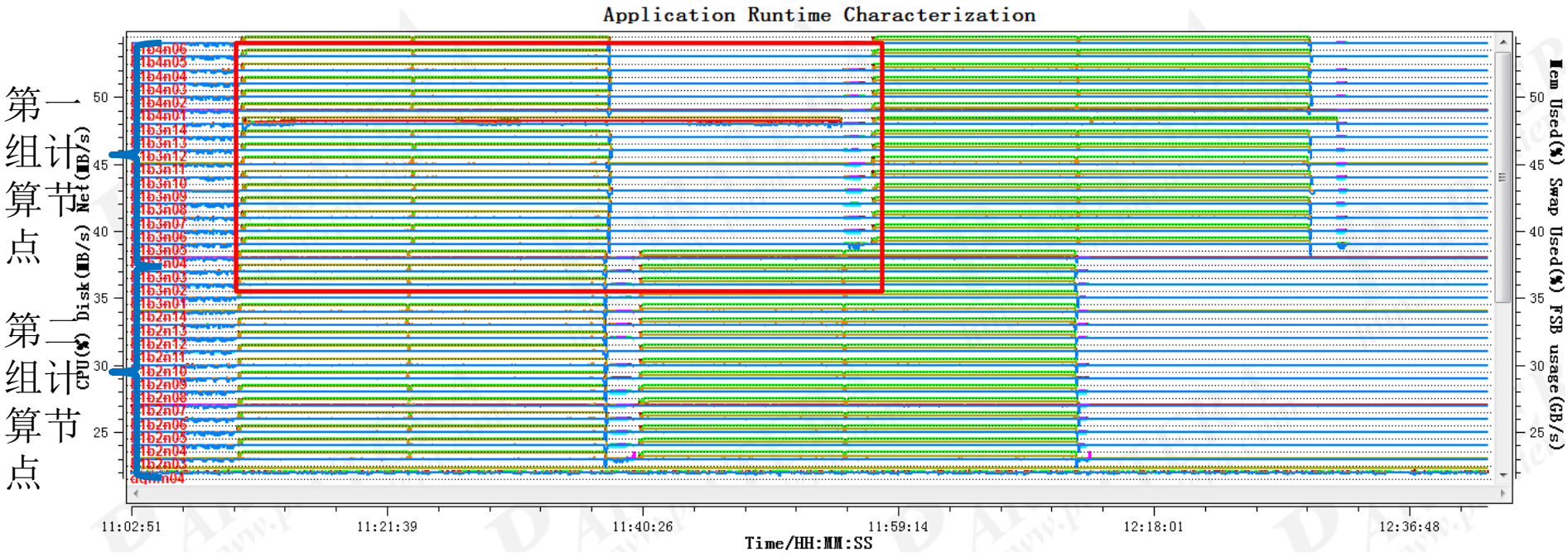
- 案例2——叠前时间偏移模块

File Style Window Help

Info Charts Edit View Layout

BaseInfo | MetricsInfo | CurrentAverage

NodeName	CPU%	CPU(sys)%	Memory%	Swap%	Net Send(MB/	Net Recv(MB/	Disk Read(MB	Disk Write(ME	IB Send(MB/s	IB Recv(MB/s)
h1b4n01	59.4784	0.653348	40.1996	0.129984	0.251802	-0.350096	0.602357	-0.595129	N/A	N/A
h1b3n14	73.255	3.73985	64.4451	20.7197	0.25192	-0.355351	2.11561	-1.90138	N/A	N/A
h1b3n13	59.425	0.684784	40.2599	0.125789	0.251596	-0.349812	0.61413	-0.534829	N/A	N/A
h1b3n12	59.3815	0.66197	40.3801	0.128357	0.257539	-0.570728	0.645483	-0.54743	N/A	N/A

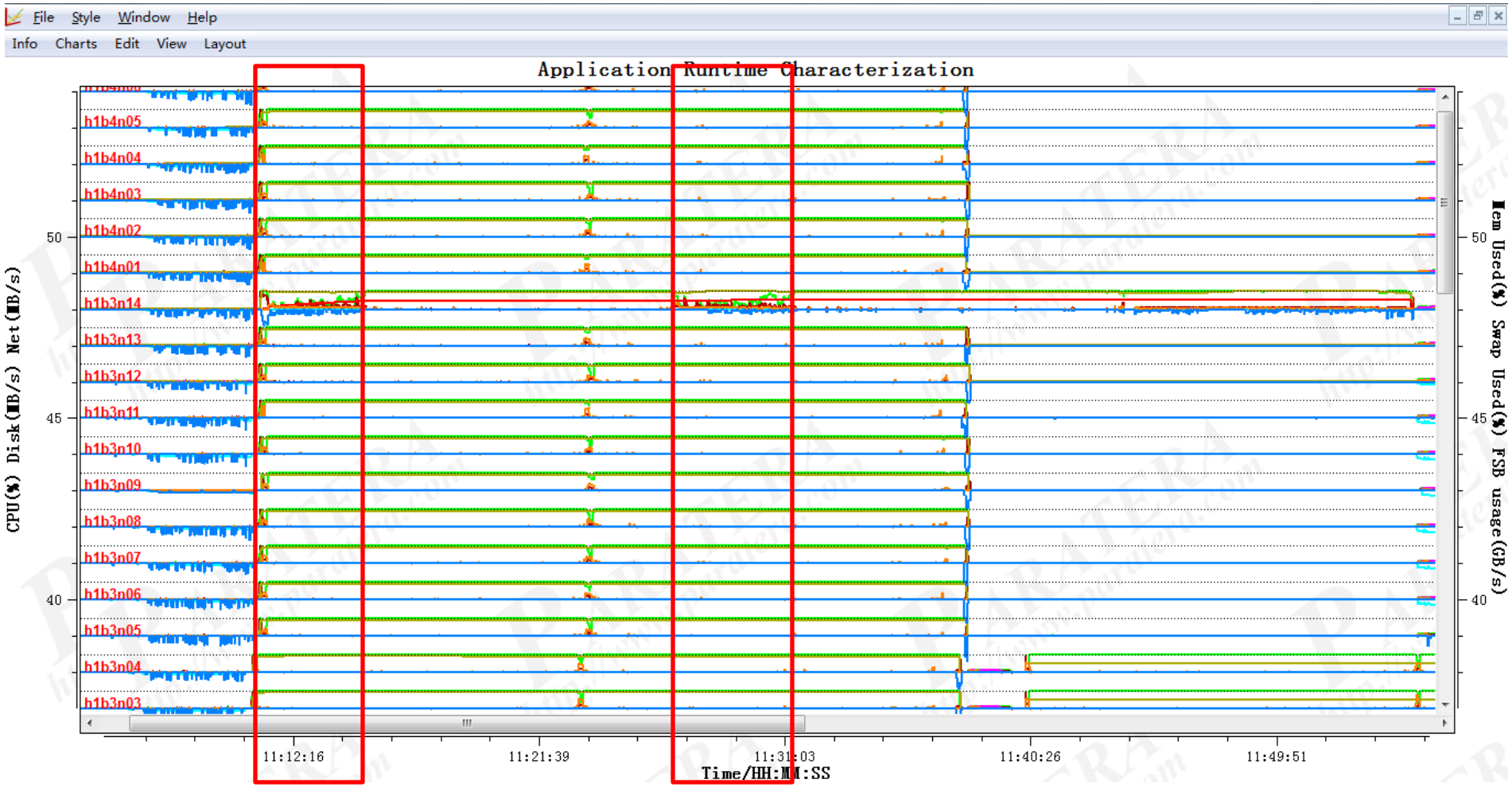


第一组计算节点

第二组计算节点

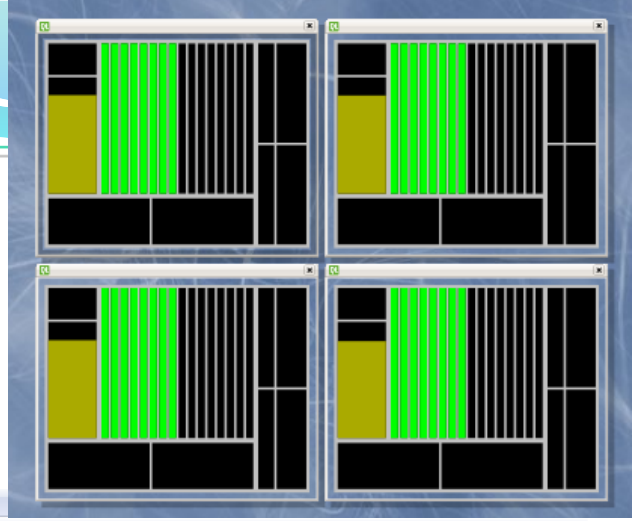
Paratune使用案例

- 案例2——叠前时间偏移模块



Paratune使用案例

- 案例3——Para+基准测试之Linpack测试

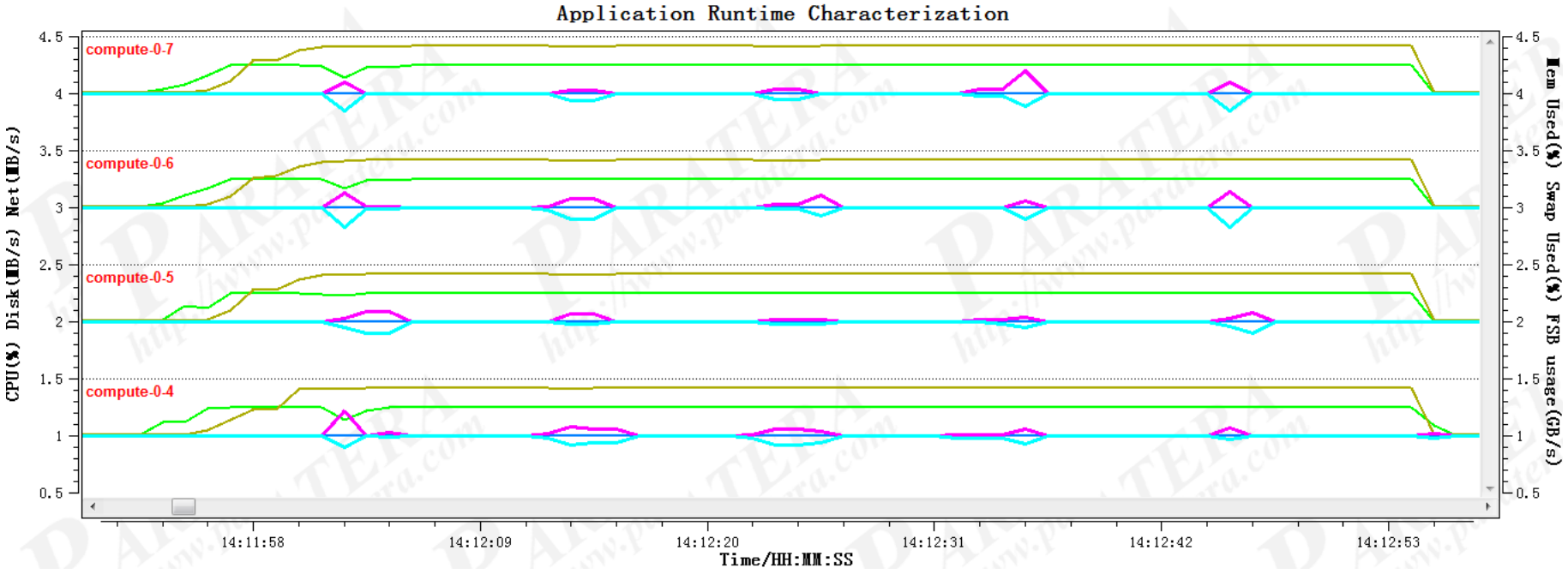


File Style Window Help

Info Charts Edit View Layout

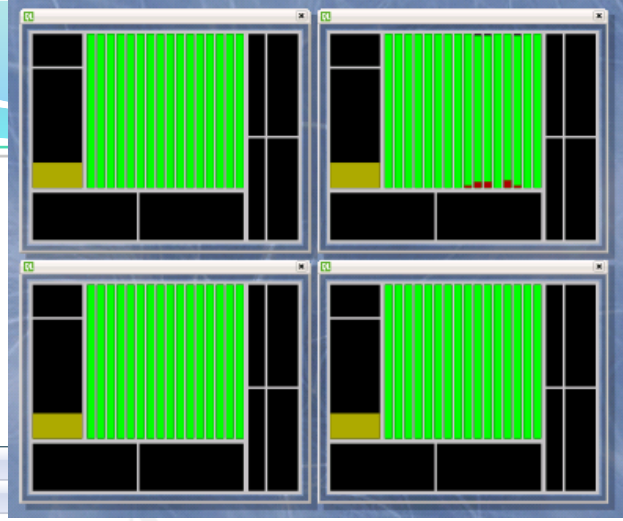
BaseInfo | MetricsInfo | CurrentAverage

NodeName	CPU%	CPU(sys)%	Memory%	Swap%	Net Send(MB/	Net Recv(MB/	Disk Read(MB	Disk Write(ME	IB Send(MB/s	IB Recv(MB/s)
compute-0-4	44.9082	2.58116	70.9189	0	0.0151343	-0.00114997	0	-0.0338115	2.63378	-2.51867
compute-0-5	44.3658	3.47643	70.6503	0	0.0149838	-0.000914136	0.00160092	-0.0272797	1.91384	-1.88964
compute-0-6	44.2847	2.5001	70.4433	0	0.028968	-0.000822099	0.00160092	-0.0355405	2.22093	-2.4798
compute-0-7	43.8928	2.66781	70.4994	0	0.0149295	-0.000815798	0.00224129	-0.0251025	2.07455	-2.16003



Paratune使用案例

- 案例3——Para+基准测试之Stream测试

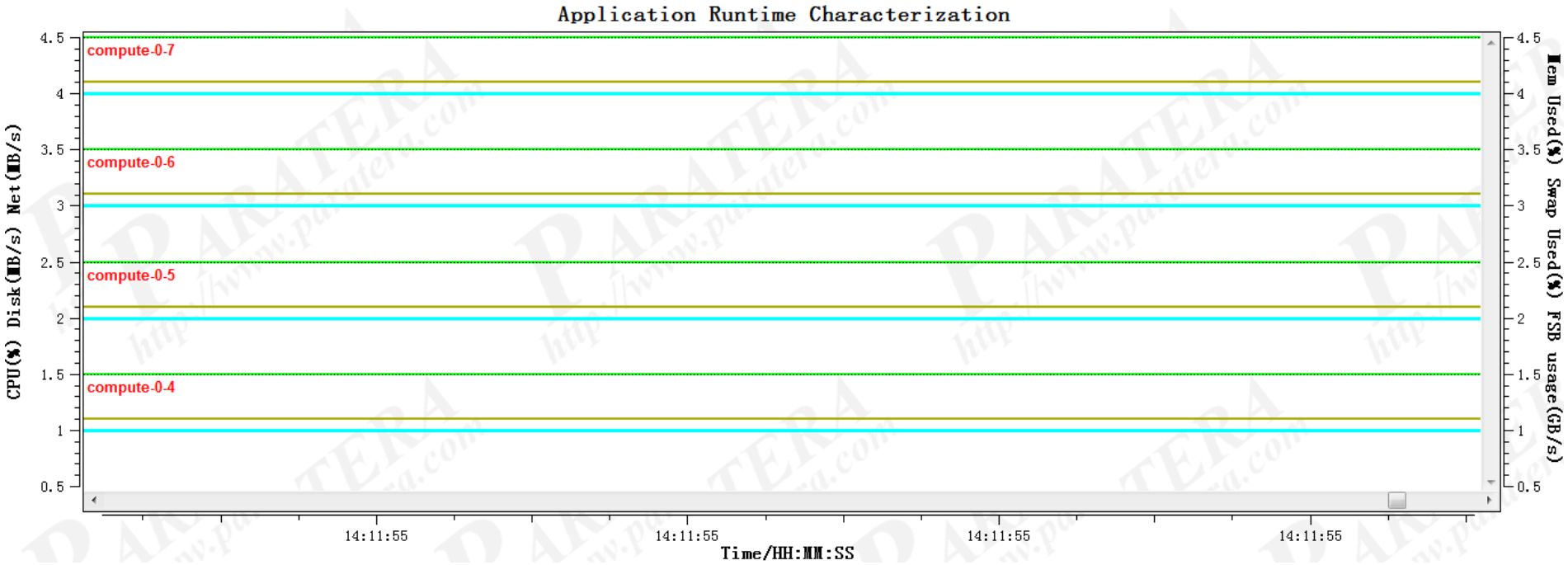


File Style Window Help

Info Charts Edit View Layout

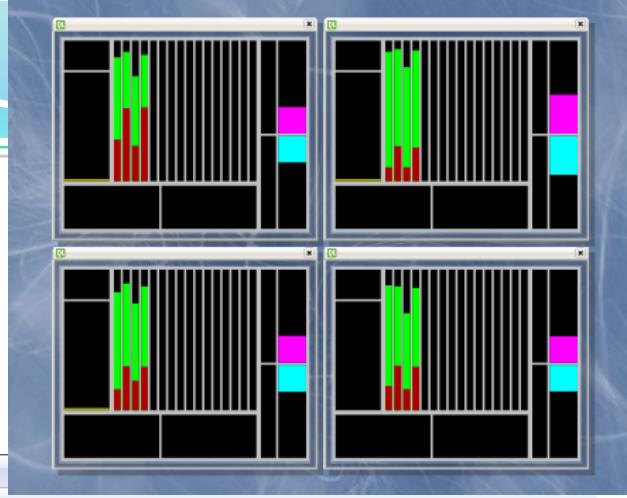
BaseInfo | MetricsInfo | CurrentAverage

NodeName	CPU%	CPU(sys)%	Memory%	Swap%	Net Send(MB/	Net Recv(MB/	Disk Read(MB	Disk Write(ME	IB Send(MB/s	IB Recv(MB/s)
compute-0-4	100	0.0625	20.3148	0	0.0140266	0	0	0	0	0
compute-0-5	99.5025	3.18	20.3225	0	0.0143814	-0.00012207	0	0	0	0
compute-0-6	100	0.0625	20.3247	0	0.0143814	-0.00012207	0	0	0	0
compute-0-7	100	44.1931	20.3212	0	0.0167179	-0.00101662	0	-0.0117188	0.000705719	-0.207862



Paratune使用案例

- 案例3——Para+基准测试之IMB测试



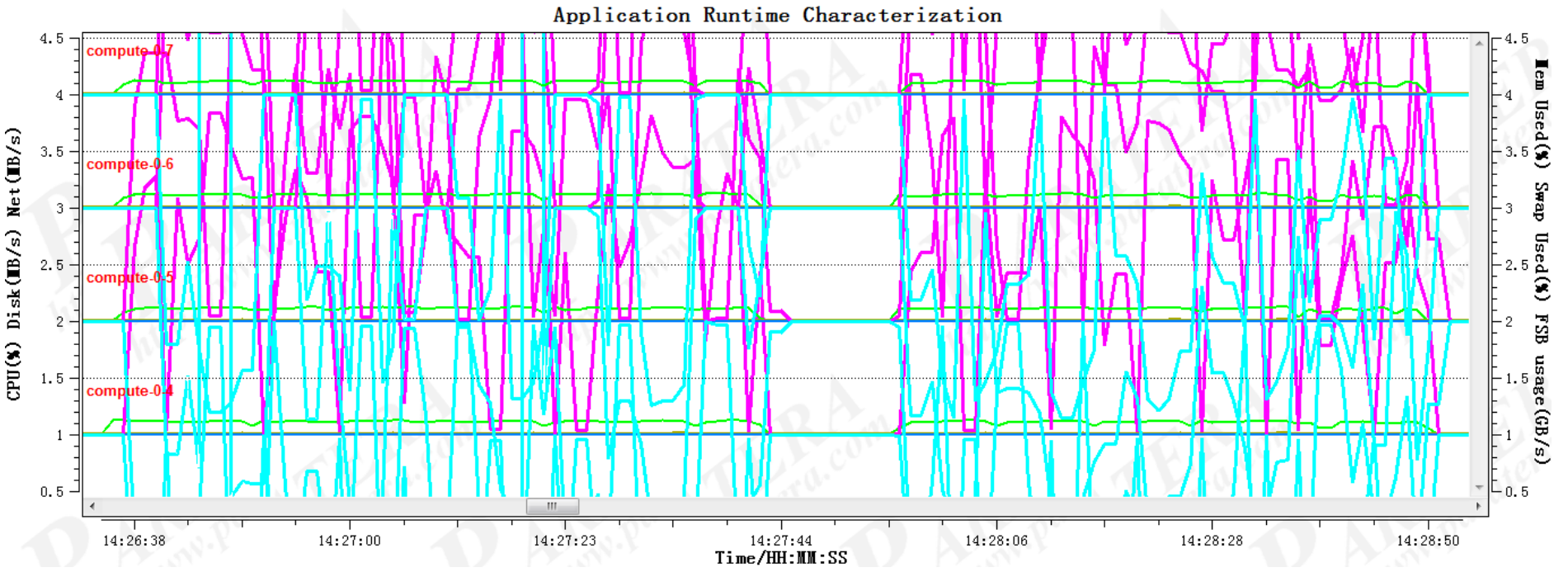
File Style Window Help

Info Charts Edit View Layout

BaseInfo | MetricsInfo | CurrentAverage

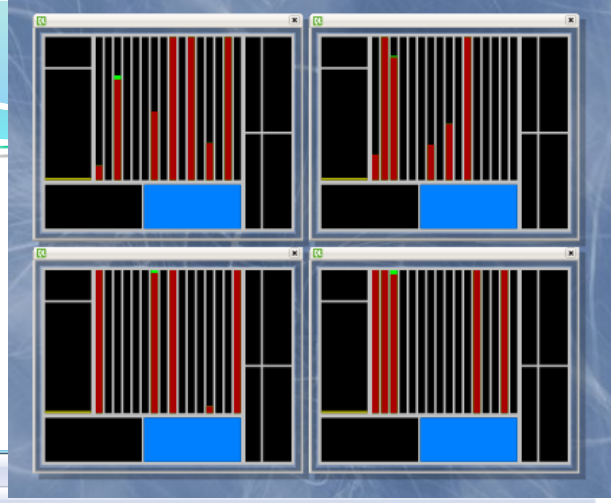
NodeName	CPU%	CPU(sys)%	Memory%	Swap%	Net Send(MB/	Net Recv(MB/	Disk Read(MB	Disk Write(M	IB Send(MB/s	IB Recv(MB/s)
compute-0-4	19.7355	8.88179	2.13513	0	0.0194179	-0.00557458	0	-0.0343628	282.024	-282.041
compute-0-5	19.287	9.14155	2.05178	0	0.0203294	-0.00611138	0	-0.0315857	273.807	-273.766
compute-0-6	19.3247	10.1479	1.81398	0	0.0340304	-0.00563095	0	-0.0363159	247.741	-250.922
compute-0-7	18.9745	10.1395	1.61731	0	0.0198622	-0.00553357	0	-0.0337219	234.061	-234.03

InfiniBand（简称IB）网络收发速率



Paratune使用案例

- 案例3——Para+基准测试之IOzone测试



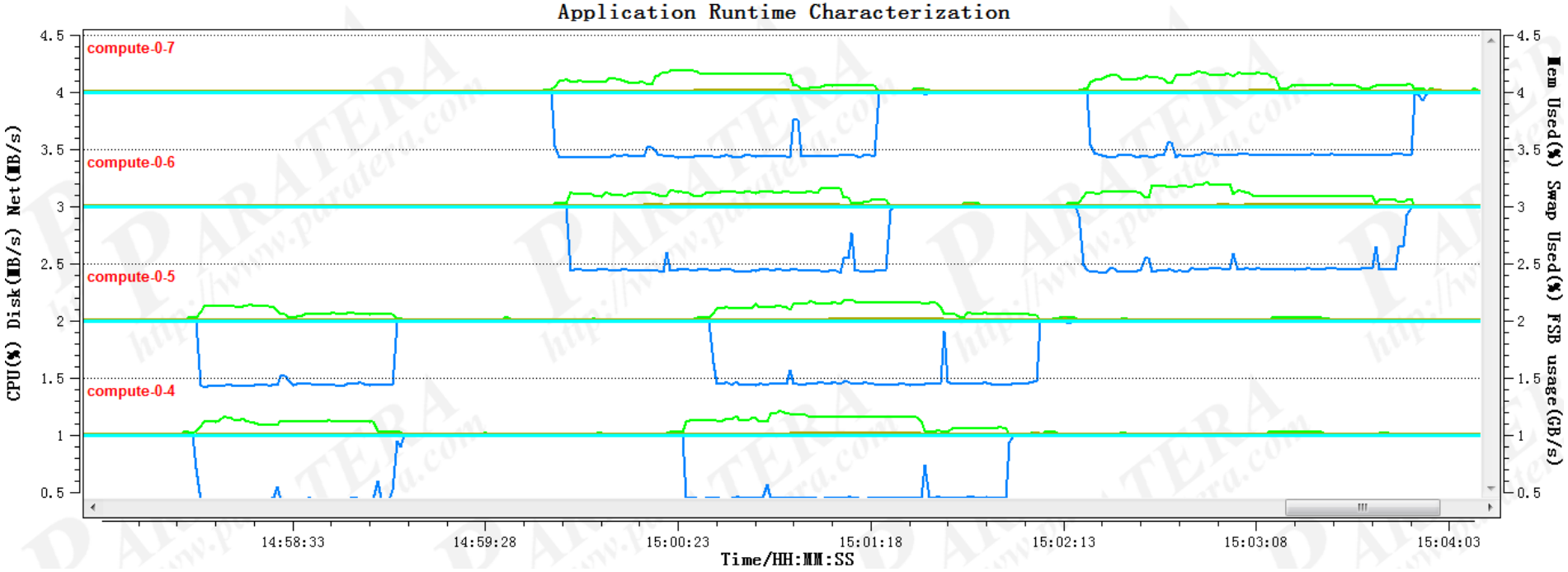
File Style Window Help

Info Charts Edit View Layout

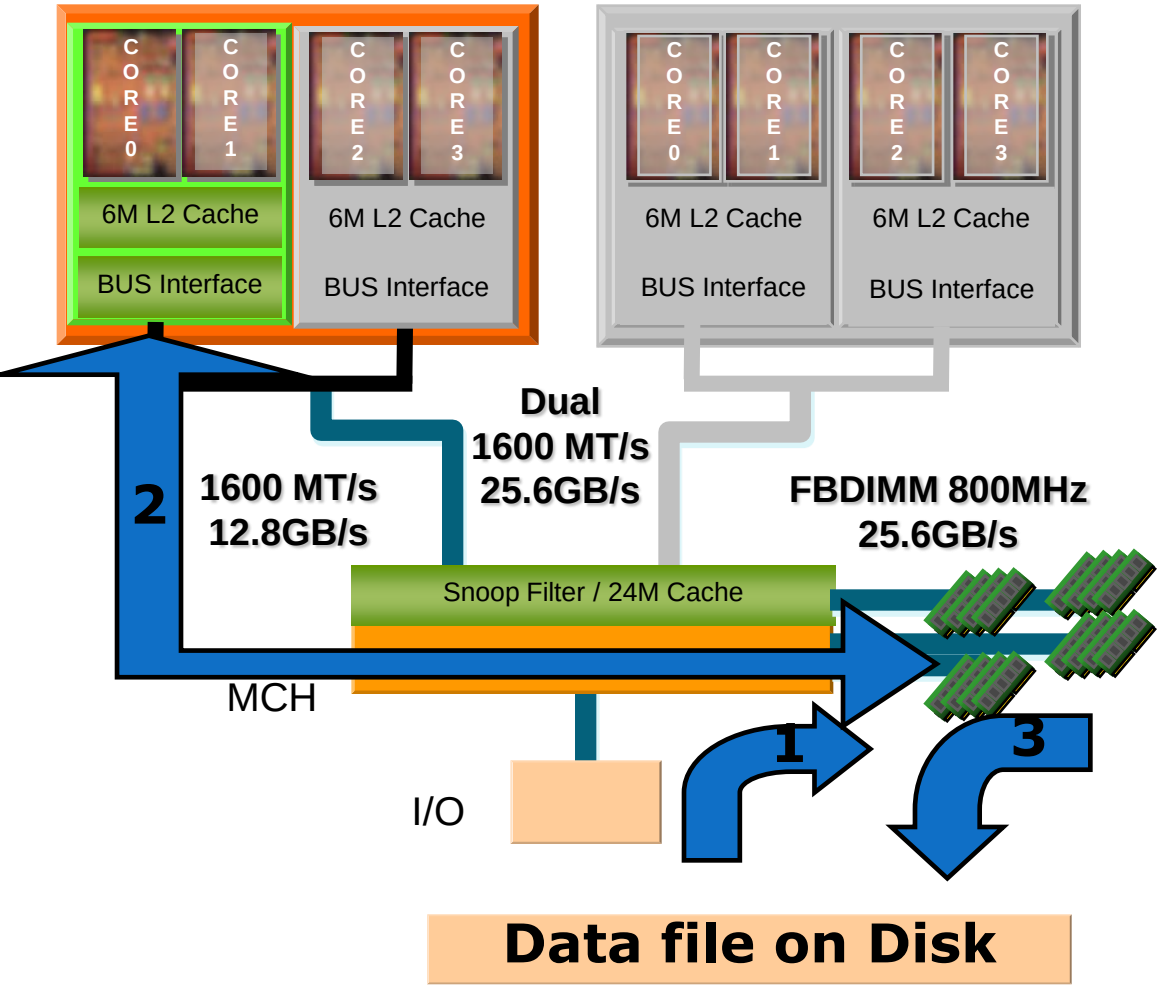
BaseInfo MetricsInfo CurrentAverage

磁盘写速率

NodeName	CPU%	CPU(sys)%	Memory%	Swap%	Net Send(MB/	Net Recv(MB/	Disk Read(MB	Disk Write(MB	IB Send(MB/s	IB Recv(MB/s)
compute-0-4	9.97471	9.93499	2.3625	0	0.0141238	-8.87638e-05	0.00138962	-41.7408	9.0559e-07	-1.35438e-06
compute-0-5	8.85198	8.80072	2.41746	0	0.0140898	-5.22678e-05	0.00134585	-41.5837	1.09793e-06	-1.28225e-06
compute-0-6	11.3062	11.2367	2.27377	0	0.0280957	-1.69417e-05	0.00165222	-50.9888	2.19051e-07	-1.22348e-06
compute-0-7	10.6945	10.6466	2.10463	0	0.0141408	-3.28924e-05	0.00187106	-51.4806	1.92338e-07	-4.56802e-07



串行程序基本运行模型/Frequency Scaling



1. 初始化
   开辟内存空间
   从硬盘读初始数据到内存

2. 迭代计算
   数据从Memory载入Cache
   指令所处理数据载入寄存器
   最终计算由计算单元完成
   数据通过Cache写回内存

3. 计算结束
   输出结果
   数据文件输出到硬盘

如何提高串行程序的运行速度（原理）

- 提高代码效率
  - 降低CPI（Cycles per Instruction），性能可以持续提高
- 实现指令级向量化操作
  - SSE (128bit Vector Instruction) by Intel Harpertown/Nehalem
    - 双精度计算性能最大可以提高2倍
    - 单精度计算性能最大可以提高4倍
  - AVX (256bit Vector Instruction) by Intel Sandy Bridge
  - LNI (512bit Vector Instruction) by Intel Larrabee
- 提高Cache利用率（降低Cache Miss Rate）
  - 计算性能最大可以提高>10倍
- ……
  - 更多的软件优化手段视具体程序而定

如何提高串行程序的运行速度(方法一)

- 尽可能使用最新版的Intel编译器
  - Intel Fortran Composer 12.0
  - Intel C/C++ Composer 12.0
  - 使用编译器选项
    - 针对通用的性能优化： -O3 –fast
    - 针对快速除法和开方： -no-prec-div / -no-prec-sqrt
    - 针对向量化： -xS, –xT(Core) –xP(NetBurst)
    - 针对Cache Miss： -fno-alias ……
    - 高级优化选项： –ip/-ipo –prof-gen/-prof-use
    - 其它常用选项： –static –unroll(n) -recursive

如何提高串行程序的运行速度(方法二)

- 尽可使用最新版的Intel MKL数学库(10.3)
  - 函数调用接口一致，直接连接库文件即可，开发简单
  - 由Intel资深工程师针对Intel硬件平台充分优化
  - 最大限度的避免了Cache Miss/Branch MissPrediction等性能问题
  - 最大限度的使用了处理器内部的资源，比如SSE处理单元，64位扩展寄存器等
- Intel MKL提供大多数常用数学函数
  - BLAS
  - FFT
  - ……

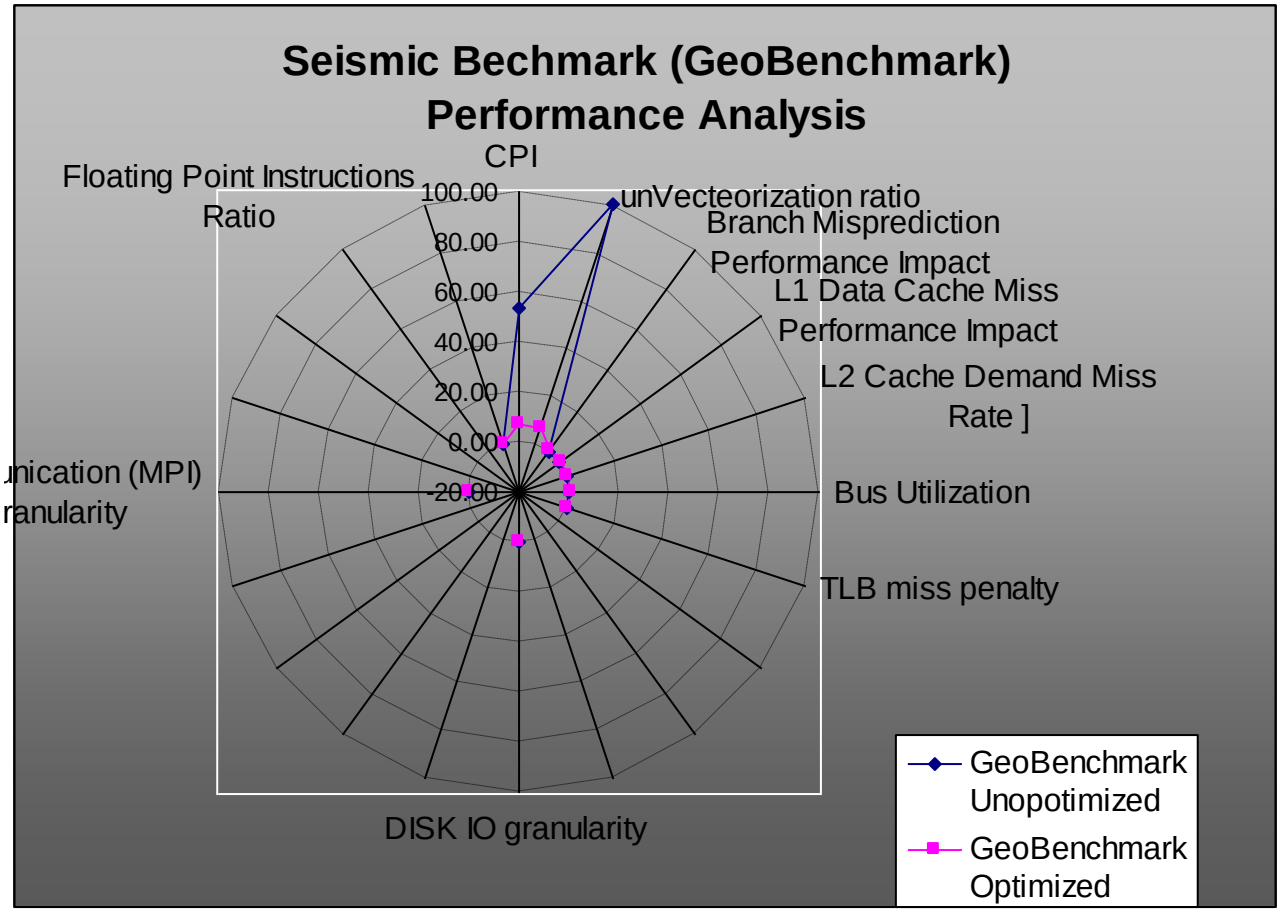
如何提高串行程序的运行速度(方法三)

- 修改应用程序辅助编译器优化程序
  - 通过Intel VTune Amplifier 查看程序运行热点函数和热点代码段
  - 修改数值计算算法
  - 修改计算机实现算法
  - 具体案例具体分析.....

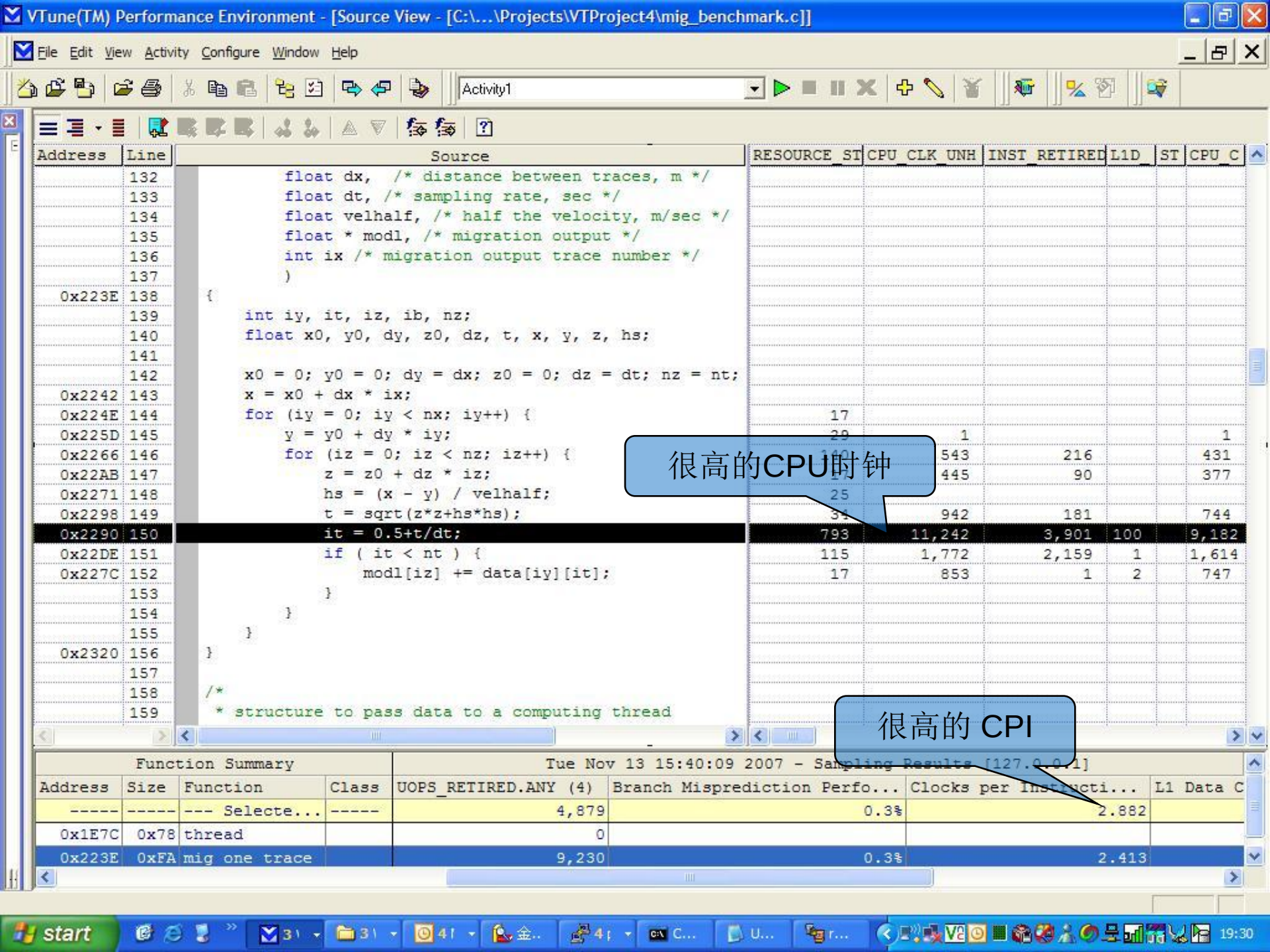
- Case Study: GeoBenchmark

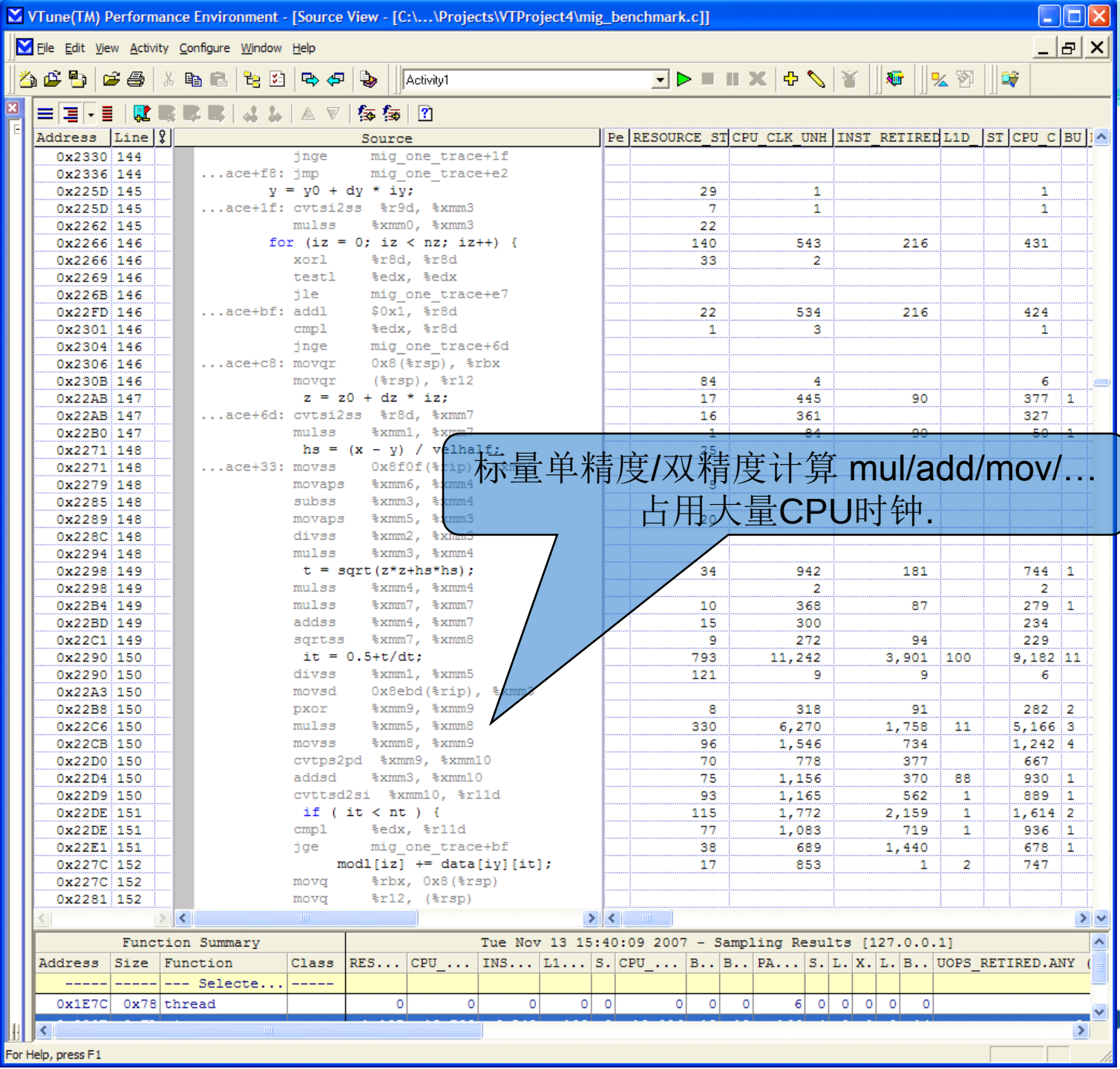
GeoBenchmark* 调优雷达

GeoBenchmark* 是由俄罗斯Evgeny Kurin开发的一套简单有效的地震资料处理基准程序:http://geocomputing.narod.ru/benchmark.html



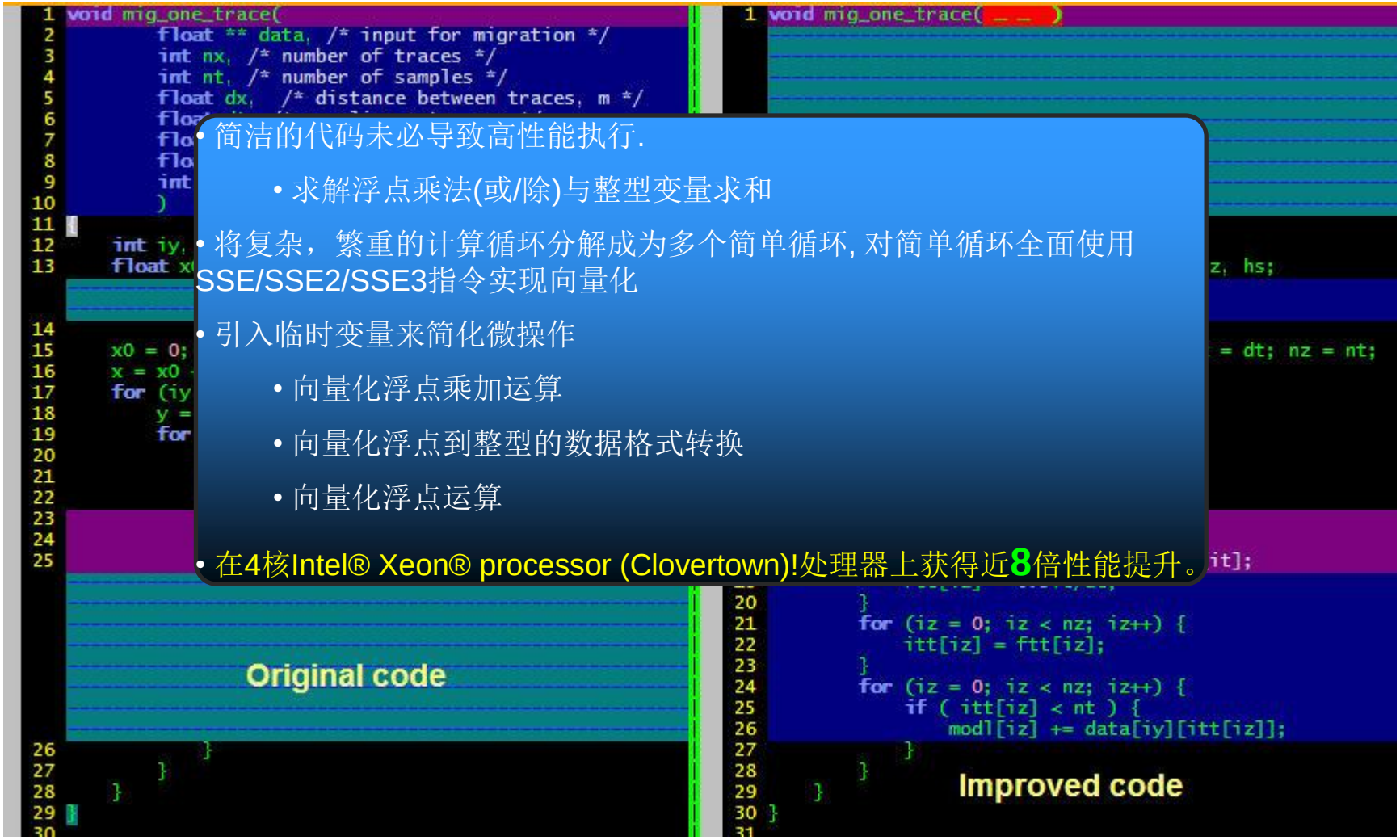
1. GeoBenchmark data is tuned by Intel CRT and accepted by Evgeny Kurin, permitted by Kurin to share here.


VTune(TM) Performance Environment - [Source View - [C:\...\Projects\VTProject4\mig_benchmark.c]]
File Edit View Activity Configure Window Help
Activity1
Address | Line | Source | RESOURCE_ST | CPU_CLK_UNH | INST_RETIRED | L1D_ | ST | CPU_C
132 float dx, /* distance between traces, m */
133 float dt, /* sampling rate, sec */
134 float velhalf, /* half the velocity, m/sec */
135 float * modl, /* migration output */
136 int ix /* migration output trace number */
137)
0x223E 138 {
139 int iy, it, iz, ib, nz;
140 float x0, y0, dy, z0, dz, t, x, y, z, hs;
141
142 x0 = 0; y0 = 0; dy = dx; z0 = 0; dz = dt; nz = nt;
0x2242 143 x = x0 + dx * ix;
0x224E 144 for (iy = 0; iy < nx; iy++) { 17
0x225D 145 y = y0 + dy * iy; 29 1 1
0x2266 146 for (iz = 0; iz < nz; iz++) { 543 216 431
0x22AB 147 z = z0 + dz * iz; 445 90 377
0x2271 148 hs = (x - y) / velhalf; 25
0x2298 149 t = sqrt(z*z+hs*hs); 942 181 744
0x2290 150 it = 0.5+t/dt; 793 11,242 3,901 100 9,182
0x22DE 151 if (it < nt) { 115 1,772 2,159 1 1,614
0x227C 152 modl[iz] += data[iy][it]; 17 853 1 2 747
153 }
154 }
155 }
0x2320 156 }
157
158 /*
159 * structure to pass data to a computing thread
很高的CPU时钟
很高的 CPI
Function Summary
Tue Nov 13 15:40:09 2007 - Sampling Results [127.0.0.1]
Address | Size | Function | Class | UOPS_RETIRED.ANY (4) | Branch Misprediction Perfo... | Clocks per Instructi... | L1 Data C
----- | ----- | --- Selecte... | ----- | 4,879 | 0.3% | 2.882
0x1E7C | 0x78 | thread | | 0 | | |
0x223E | 0xFA | mig one trace | | 9,230 | 0.3% | 2.413
start
19:30

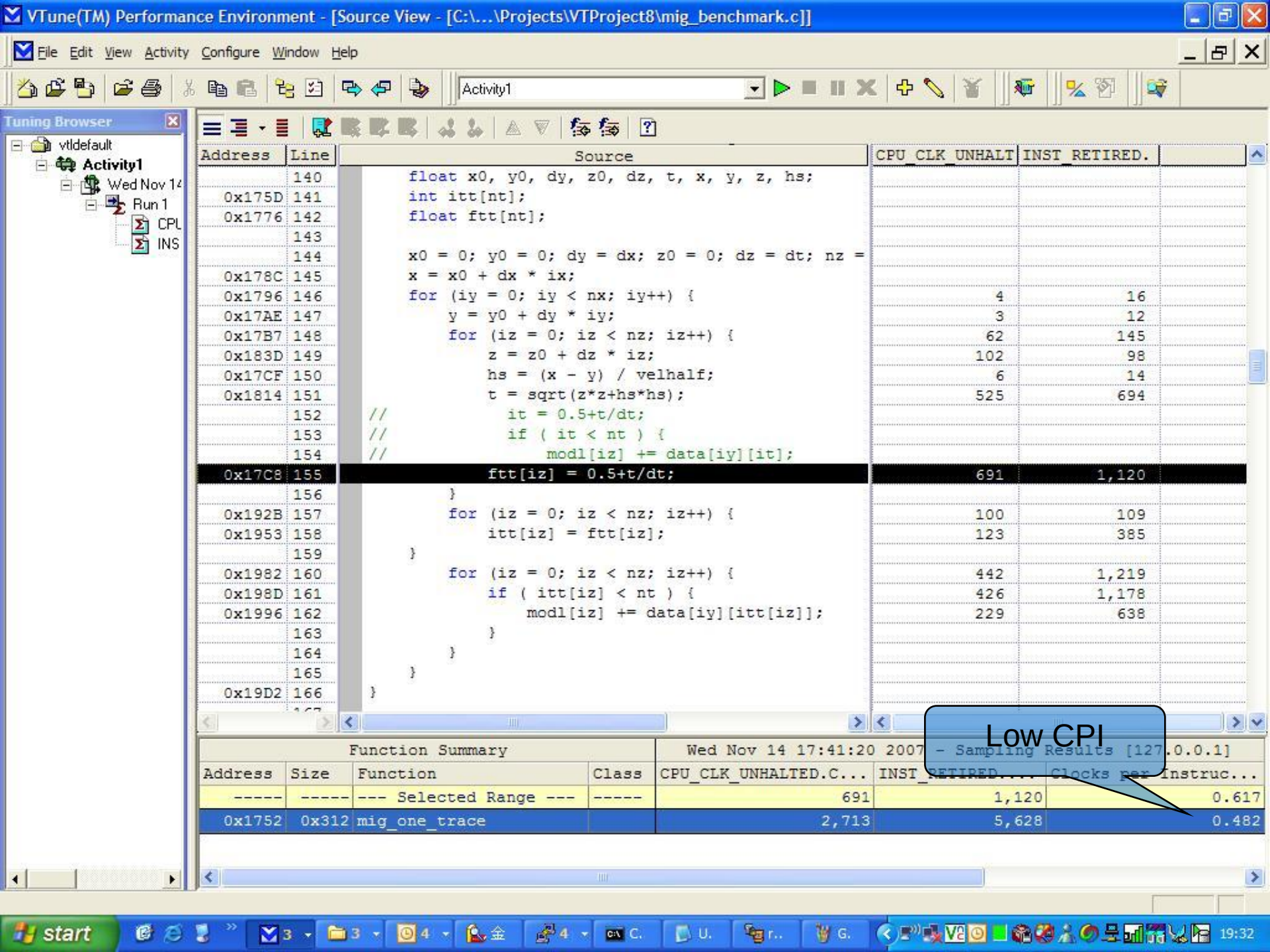


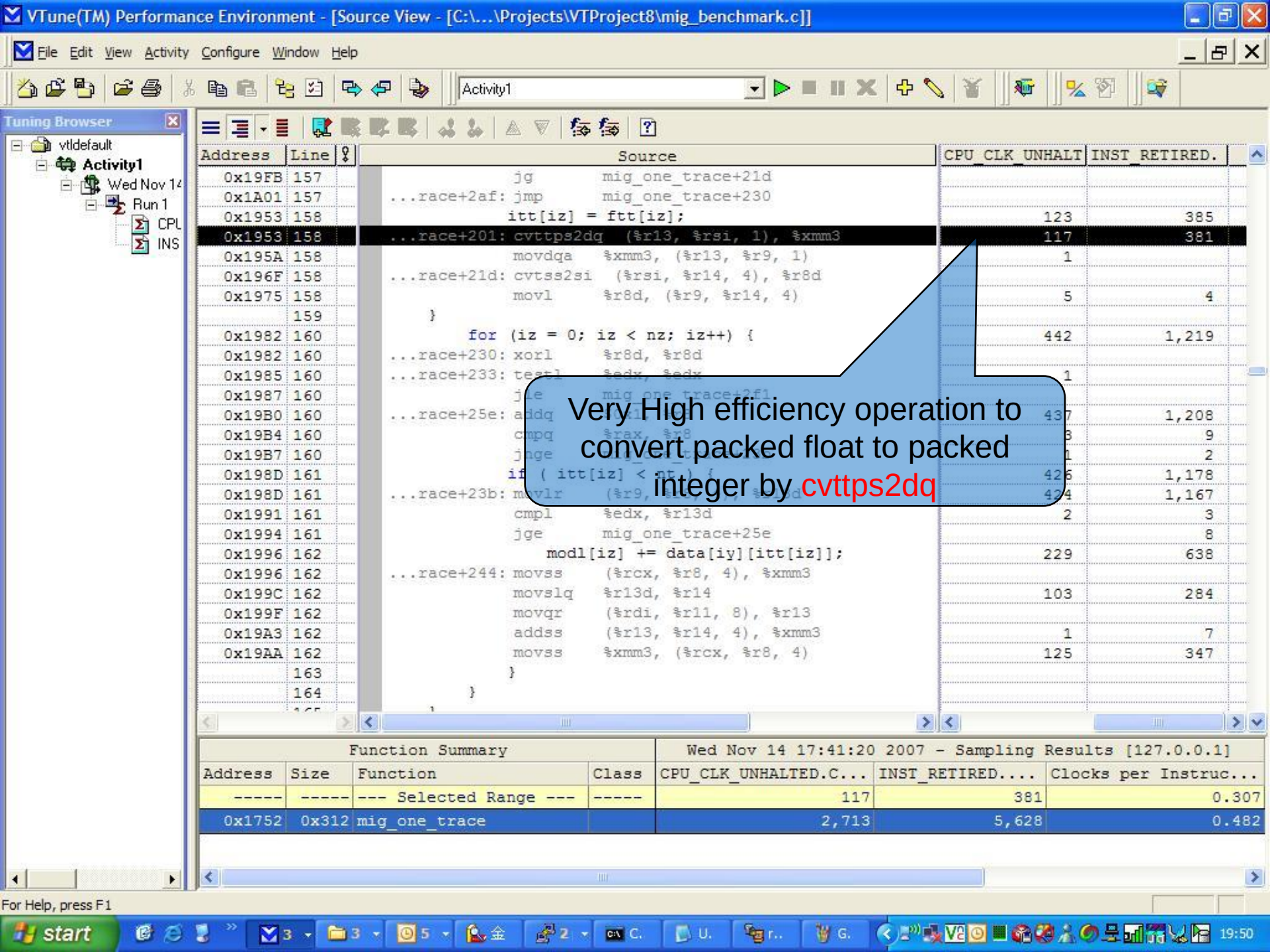
，FSB/Cache均不存在瓶颈，但代码低效：

1. 使用标量指令
2. 使用指令 cvtps2pd 转换浮点为双精度，而后使用指令 cvttsd2si 将双精度转化为整型值



1. GeoBenchmark code modification and share is granted by Kurin for Intel IDF usage.


VTune(TM) Performance Environment - [Source View - [C:\...\Projects\VTProject8\mig_benchmark.c]]
File Edit View Activity Configure Window Help
Activity1
Tuning Browser
vtldefault
Activity1
Wed Nov 14
Run 1
CPU
INS
Address Line Source CPU_CLK_UNHALT INST_RETIRED.
140 float x0, y0, dy, z0, dz, t, x, y, z, hs;
0x175D 141 int itt[nt];
0x1776 142 float ftt[nt];
143
144 x0 = 0; y0 = 0; dy = dx; z0 = 0; dz = dt; nz =
0x178C 145 x = x0 + dx * ix;
0x1796 146 for (iy = 0; iy < nx; iy++) { 4 16
0x17AE 147 y = y0 + dy * iy; 3 12
0x17B7 148 for (iz = 0; iz < nz; iz++) { 62 145
0x183D 149 z = z0 + dz * iz; 102 98
0x17CF 150 hs = (x - y) / velhalf; 6 14
0x1814 151 t = sqrt(z*z+hs*hs); 525 694
152 // it = 0.5+t/dt;
153 // if (it < nt) {
154 // modl[iz] += data[iy][it];
0x17C8 155 ftt[iz] = 0.5+t/dt; 691 1,120
156 }
0x192B 157 for (iz = 0; iz < nz; iz++) { 100 109
0x1953 158 itt[iz] = ftt[iz]; 123 385
159 }
0x1982 160 for (iz = 0; iz < nz; iz++) { 442 1,219
0x198D 161 if (itt[iz] < nt) { 426 1,178
0x1996 162 modl[iz] += data[iy][itt[iz]]; 229 638
163 }
164 }
165 }
0x19D2 166 }
Function Summary
Wed Nov 14 17:41:20 2007 - Sampling Results [127.0.0.1]
Address Size Function Class CPU_CLK_UNHALTED.C... INST_RETIRED.... Clocks per Instruc...
----- ----- --- Selected Range --- ----- 691 1,120 0.617
0x1752 0x312 mig_one_trace 2,713 5,628 0.482
Low CPI
start
19:32


VTune(TM) Performance Environment - [Source View - [C:\...\Projects\VTProject8\mig_benchmark.c]]
File Edit View Activity Configure Window Help
Activity1
Tuning Browser
vtldefault
Activity1
Wed Nov 14
Run 1
CPU
INS
Address Line Source CPU_CLK_UNHALT INST_RETIRED.
0x19FB 157 jg mig_one_trace+21d
0x1A01 157 ...race+2af: jmp mig_one_trace+230
0x1953 158 itt[iz] = ftt[iz]; 123 385
0x1953 158 ...race+201: cvttps2dq (%r13, %rsi, 1), %xmm3 117 381
0x195A 158 movdqa %xmm3, (%r13, %r9, 1) 1
0x196F 158 ...race+21d: cvtss2si (%rsi, %r14, 4), %r8d
0x1975 158 movl %r8d, (%r9, %r14, 4) 5 4
159 }
0x1982 160 for (iz = 0; iz < nz; iz++) { 442 1,219
0x1982 160 ...race+230: xorl %r8d, %r8d
0x1985 160 ...race+233: testl %edx, %edx 1
0x1987 160 jle mig_one_trace+2f1
0x19B0 160 ...race+25e: addq 437 1,208
0x19B4 160 cmpq %rax, %r8 3 9
0x19B7 160 jne 1 2
0x198D 161 if (itt[iz] < ftt[iz]) { 426 1,178
0x198D 161 ...race+23b: movlr (%r9, 424 1,167
0x1991 161 cmpl %edx, %r13d 2 3
0x1994 161 jge mig_one_trace+25e 8
0x1996 162 modl[iz] += data[iy][itt[iz]]; 229 638
0x1996 162 ...race+244: movss (%rcx, %r8, 4), %xmm3
0x199C 162 movslq %r13d, %r14 103 284
0x199F 162 movqr (%rdi, %r11, 8), %r13
0x19A3 162 addss (%r13, %r14, 4), %xmm3 1 7
0x19AA 162 movss %xmm3, (%rcx, %r8, 4) 125 347
163 }
164 }
Very High efficiency operation to convert packed float to packed integer by cvttps2dq
Function Summary
Wed Nov 14 17:41:20 2007 - Sampling Results [127.0.0.1]
Address Size Function Class CPU_CLK_UNHALTED.C... INST_RETIRED.... Clocks per Instruc...
----- ----- --- Selected Range --- ----- 117 381 0.307
0x1752 0x312 mig_one_trace 2,713 5,628 0.482
For Help, press F1
start
19:50

CVTTPS2DQ--Convert Packed Single-Precision Floating-Point Values to Packed Doubleword Integers with Truncation

Opcode	Instruction	Description
F3 0F 5B /r	CVTTPS2DQ *xmm1, xmm2/m128*	Convert four single-precision floating-point values from *xmm2/m128* to four signed doubleword integers in *xmm1* using truncation.

Converts four packed single-precision floating-point values in the source operand (second operand) to four packed signed doubleword integers in the destination operand (first operand). The source operand can be an XMM register or a 128-bit memory location. The destination operand is an XMM register.

When a conversion is inexact, a truncated result is returned. If a converted result is larger than the maximum signed doubleword integer, the floating-point invalid exception is raised, and if this exception is masked, the indefinite integer value (80000000H) is returned.

Operation

```
DEST[31-0] ← Convert_Single_Precision_Floating_Point_To_Integer_Truncate(SRC[31-0]);
DEST[63-32] ← Convert_Single_Precision_Floating_Point_To_Integer_Truncate(SRC[63-32]);
DEST[95-64] ← Convert_Single_Precision_Floating_Point_To_Integer_Truncate(SRC[95-64]);
DEST[127-96] ← Convert_Single_Precision_Floating_Point_To_Integer_
Truncate(SRC[127-96]);
```

Intel(R) C++ Compiler Intrinsic Equivalent

```
__m128d _mm_cvttps_epi32(__m128d a)
```

SIMD Floating-Point Exceptions

Invalid, Precision.

Protected Mode Exceptions

#GP(0) - For an illegal memory operand effective address in the CS, DS, ES, FS or GS segments. If memory operand is not aligned on a 16-byte boundary, regardless of segment.

#SS(0) - For an illegal address in the SS segment.

#PF(fault-code) - For a page fault.

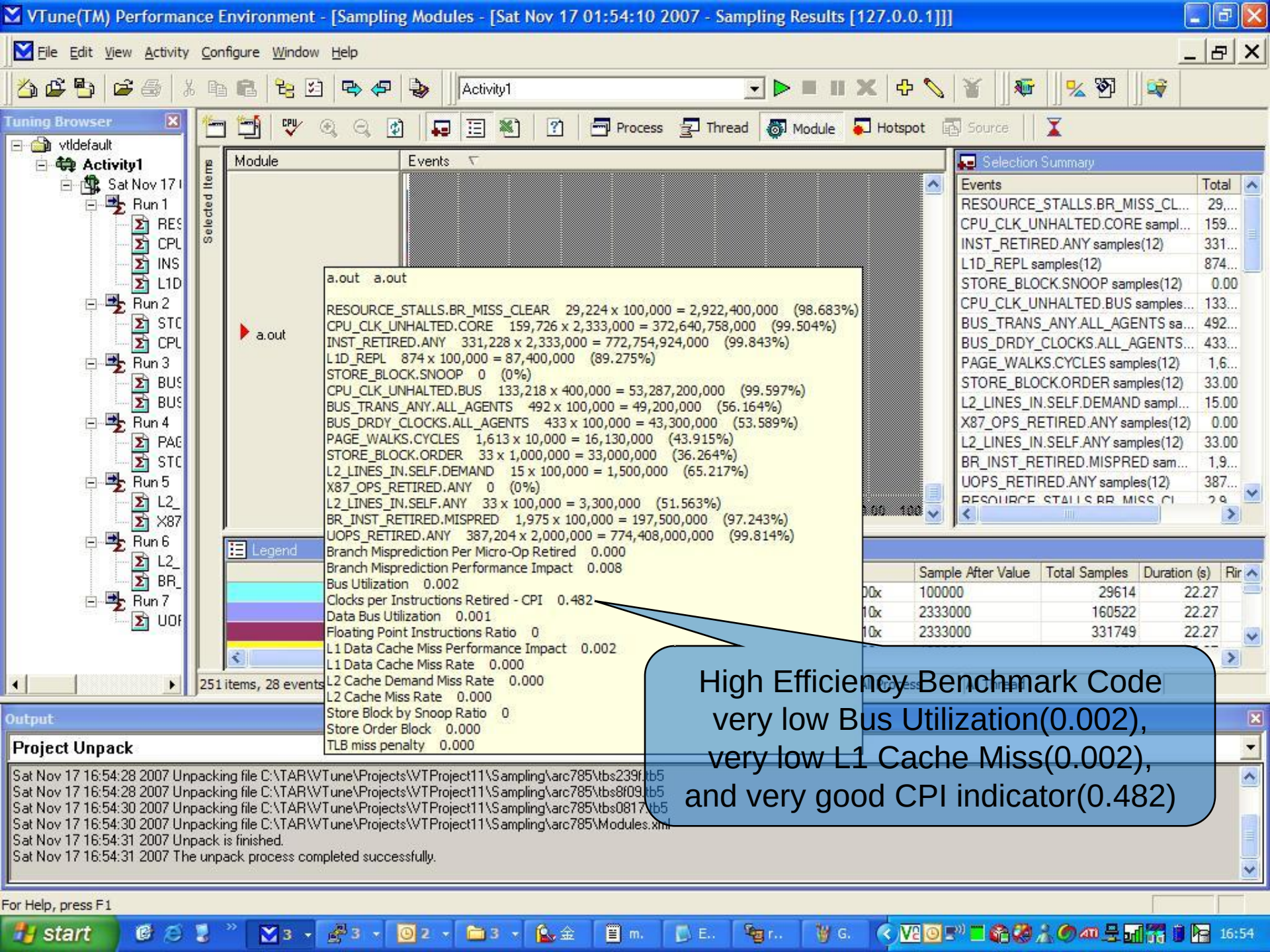
#NM - If TS in CR0 is set.

#XM - If an unmasked SIMD floating-point exception and OSXMMEXCPT in CR4 is 1.

#UD - If an unmasked SIMD floating-point exception and OSXMMEXCPT in CR4 is 0. If EM in CR0 is set. If OSFXSR in CR4 is 0. If CPUID feature flag SSE2 is 0.

Real-Address Mode Exceptions

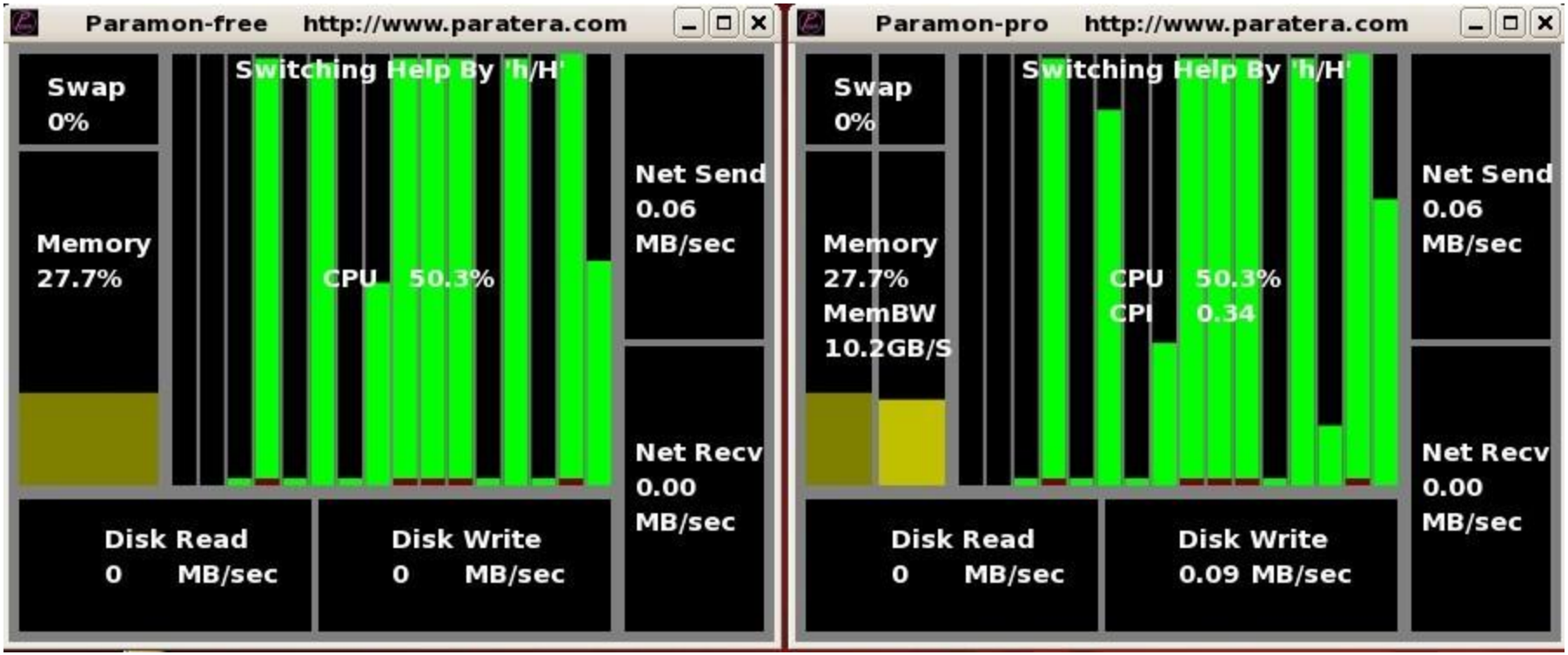
#GP(0) - If memory operand is not aligned on a 16-byte boundary, regardless of segment



High Efficiency Benchmark Code
very low Bus Utilization(0.002),
very low L1 Cache Miss(0.002),
and very good CPI indicator(0.482)

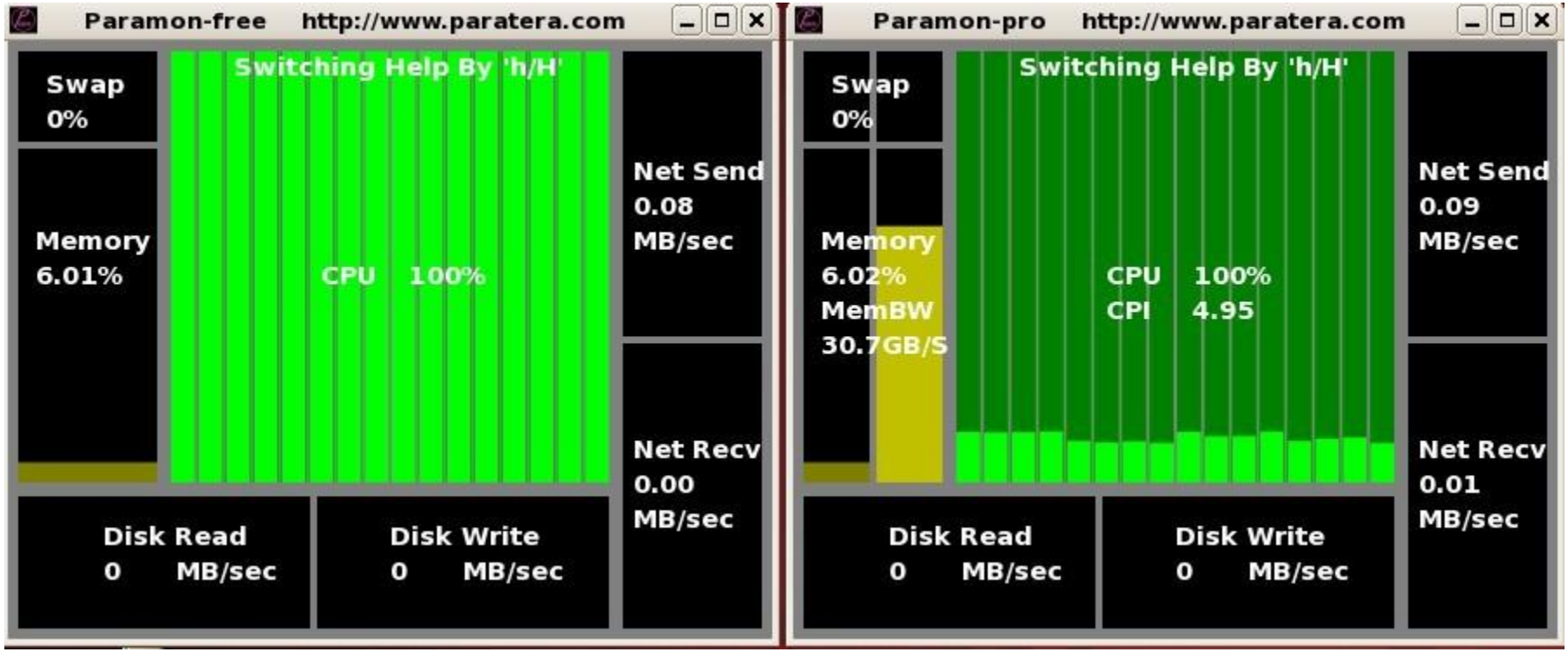
微架构级应用特征分析(单节点)
快速判断应用程序微架构运行特征

Paramon-pro单机版的主界面 – Linpack Benchmark



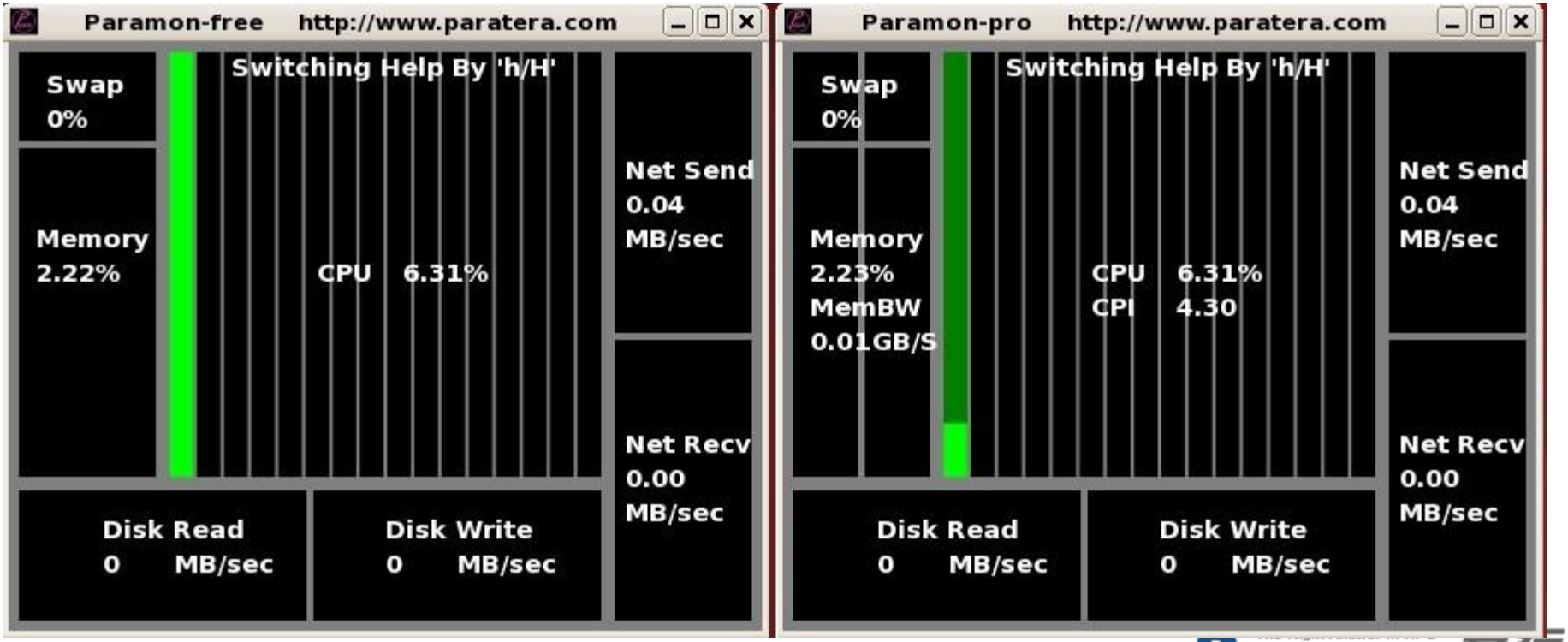
微架构级应用特征分析(单节点)
快速判断应用程序微架构运行特征

Paramon-pro单机版的主界面 - Stream Benchmark



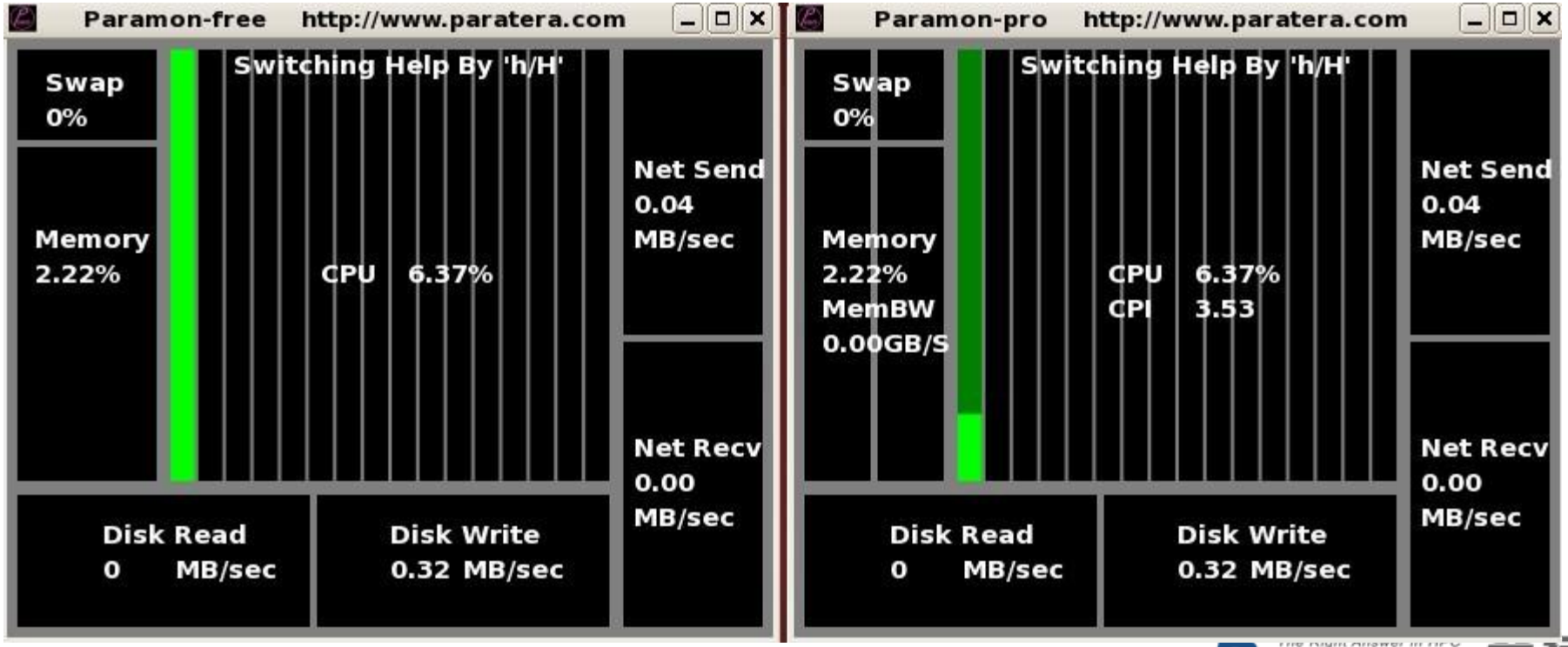
微架构级应用特征分析(单节点)
快速判断应用程序计算部分是否有优化潜力

BKMs	Description	time (s)	speedup
Baseline:	***icc 11.0.081 -lpthread ./a.out 1001 1000 1***	***14.99***	
BKM #1	*icc 11.0.081 -lpthread -no-prec-div*	***11.1***	***35.05%***
BKM #2	*_cproc_p_11.0.081 + itt[]*	***4.2***	***356.9%***
BKM #3	*_cproc_p_11.0.081 + itt[] -lpthread -no-prec-div -O3*	**3.58**	***418.7%***



微架构级应用特征分析(单节点)
快速判断应用程序计算部分是否有优化潜力

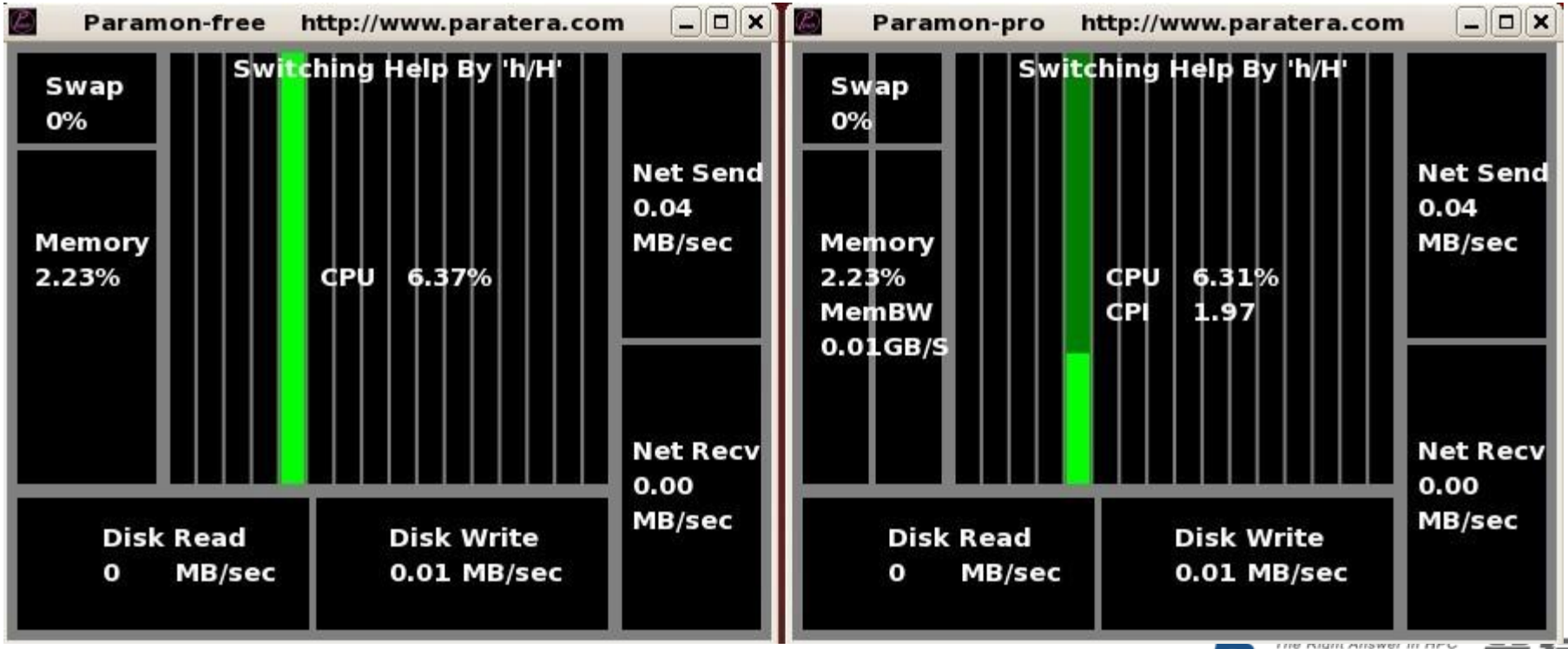
BKMs	Description	time (s)	speedup
Baseline:	***icc 11.0.081 -lpthread ./a.out 1001 1000 1***	***14.99***	
BKM #1	*icc 11.0.081 -lpthread -no-prec-div*	***11.1***	***35.05%***
BKM #2	*_cproc_p_11.0.081 + itt[]*	***4.2***	***356.9%***
BKM #3	*_cproc_p_11.0.081 + itt[] -lpthread -no-prec-div -O3*	**3.58**	***418.7%***



PARATERA 并行

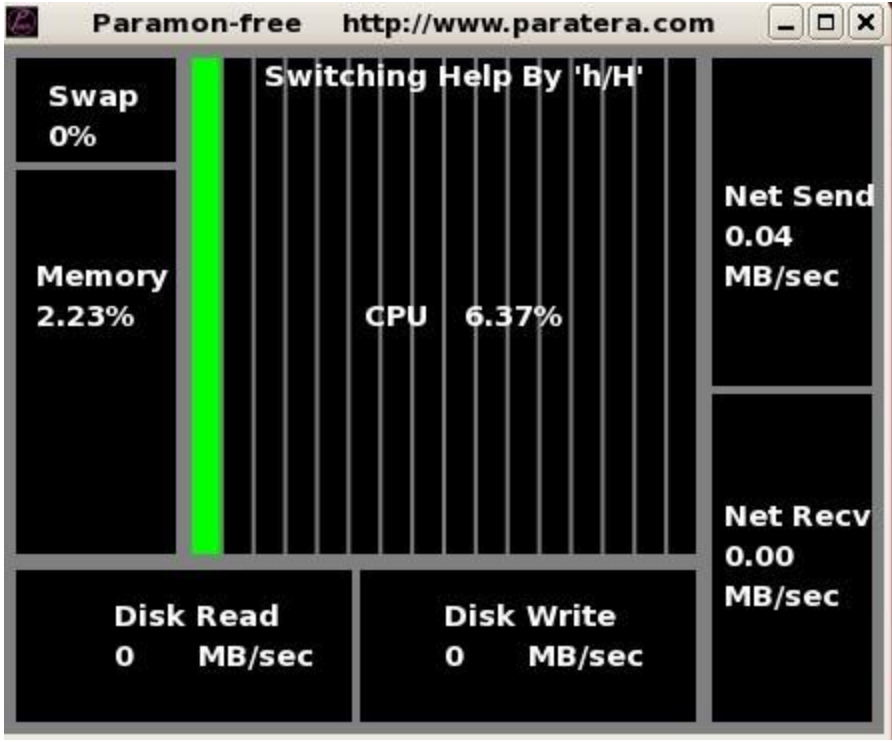
微架构级应用特征分析(单节点)
快速判断应用程序计算部分是否有优化潜力

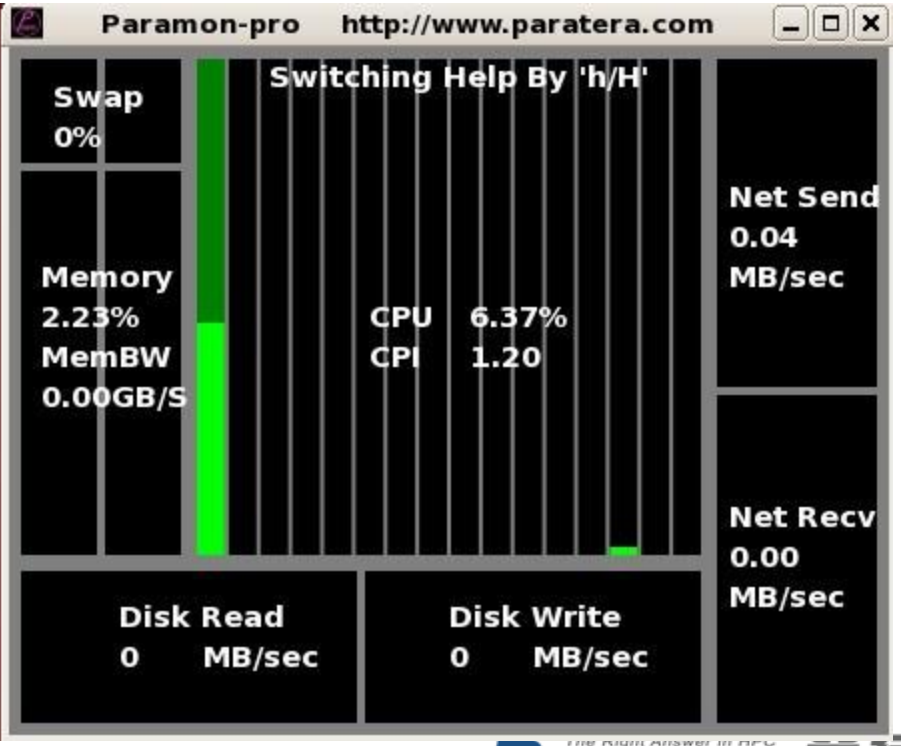
BKMs	Description	time (s)	speedup
Baseline:	***icc 11.0.081 -lpthread ./a.out 1001 1000 1***	***14.99***	
BKM #1	*icc 11.0.081 -lpthread -no-prec-div*	***11.1***	***35.05%***
BKM #2	*_cproc_p_11.0.081 + itt[]*	***4.2***	***356.9%***
BKM #3	*_cproc_p_11.0.081 + itt[] -lpthread -no-prec-div -O3*	**3.58**	***418.7%***



微架构级应用特征分析(单节点)
快速判断应用程序计算部分是否有优化潜力

BKMs	Description	time (s)	speedup
Baseline:	***icc 11.0.081 -lpthread ./a.out 1001 1000 1***	***14.99***	
BKM #1	*icc 11.0.081 -lpthread -no-prec-div*	***11.1***	***35.05%***
BKM #2	*_cproc_p_11.0.081 + itt[]*	***4.2***	***356.9%***
BKM #3	*_cproc_p_11.0.081 + itt[] -lpthread -no-prec-div -O3*	**3.58**	***418.7%***

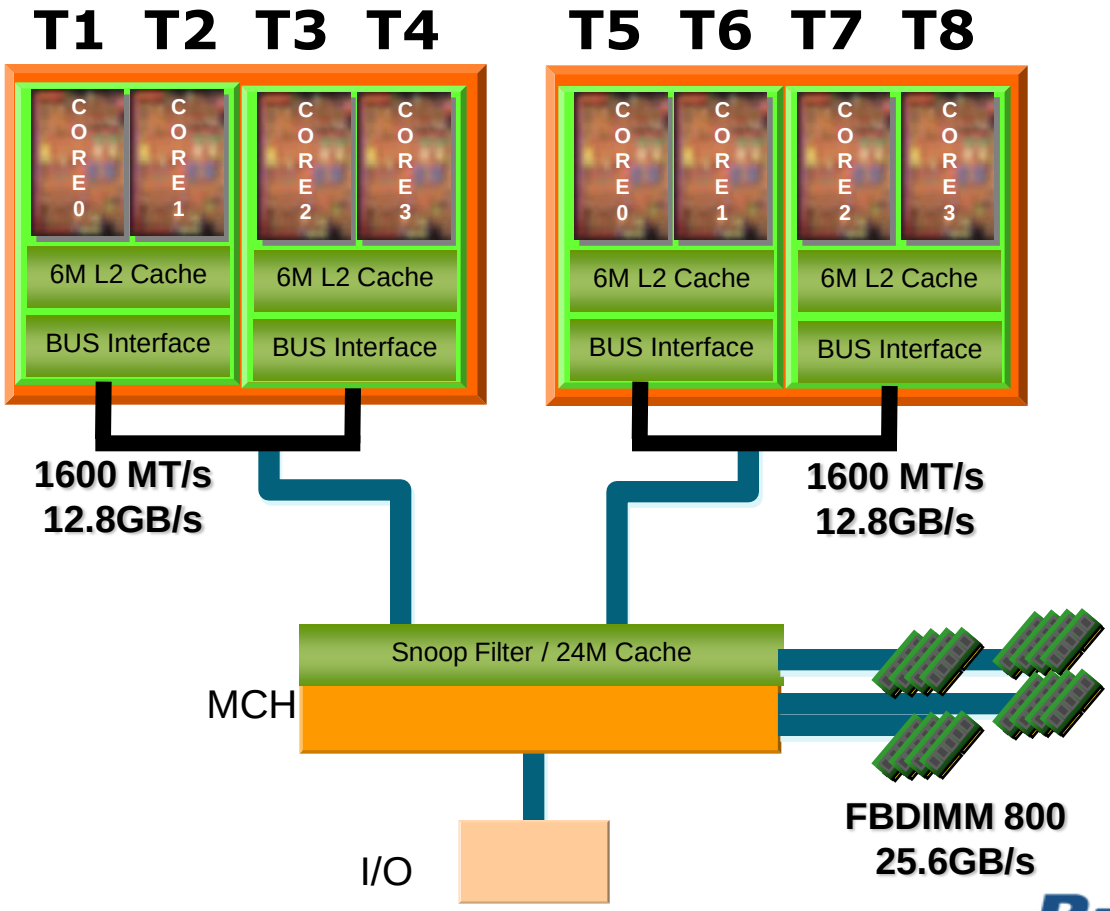




共享内存并行程序运行/Core Scaling

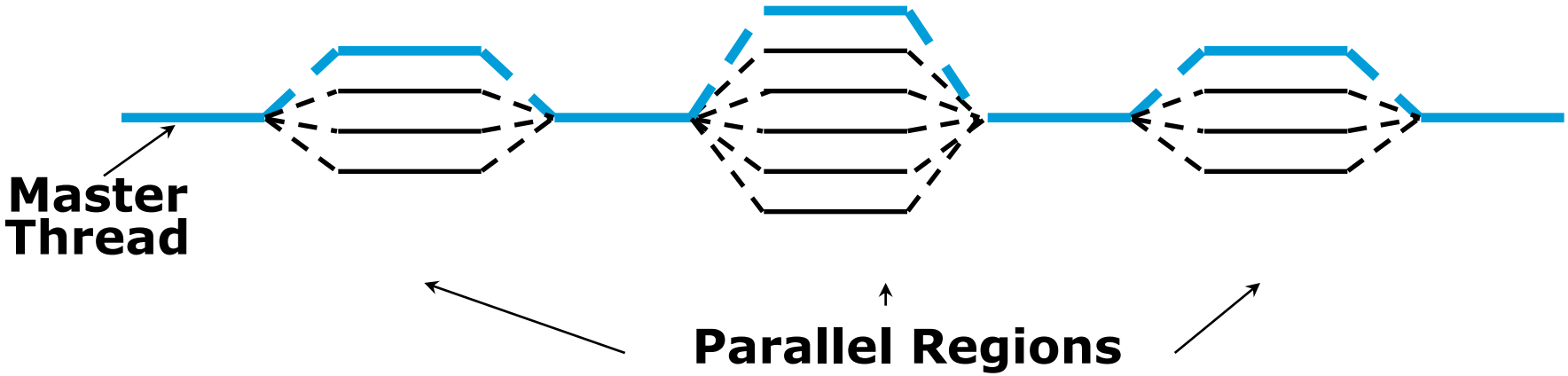
双**Socket**四核平台可以有**8**个核

使用**OpenMP**共享内存方式并行化已有的串行程序



OpenMP

- Fork-join parallelism:
  - Master thread spawns a team of threads as needed
  - Parallelism is added incrementally
    - sequential program evolves into a parallel program



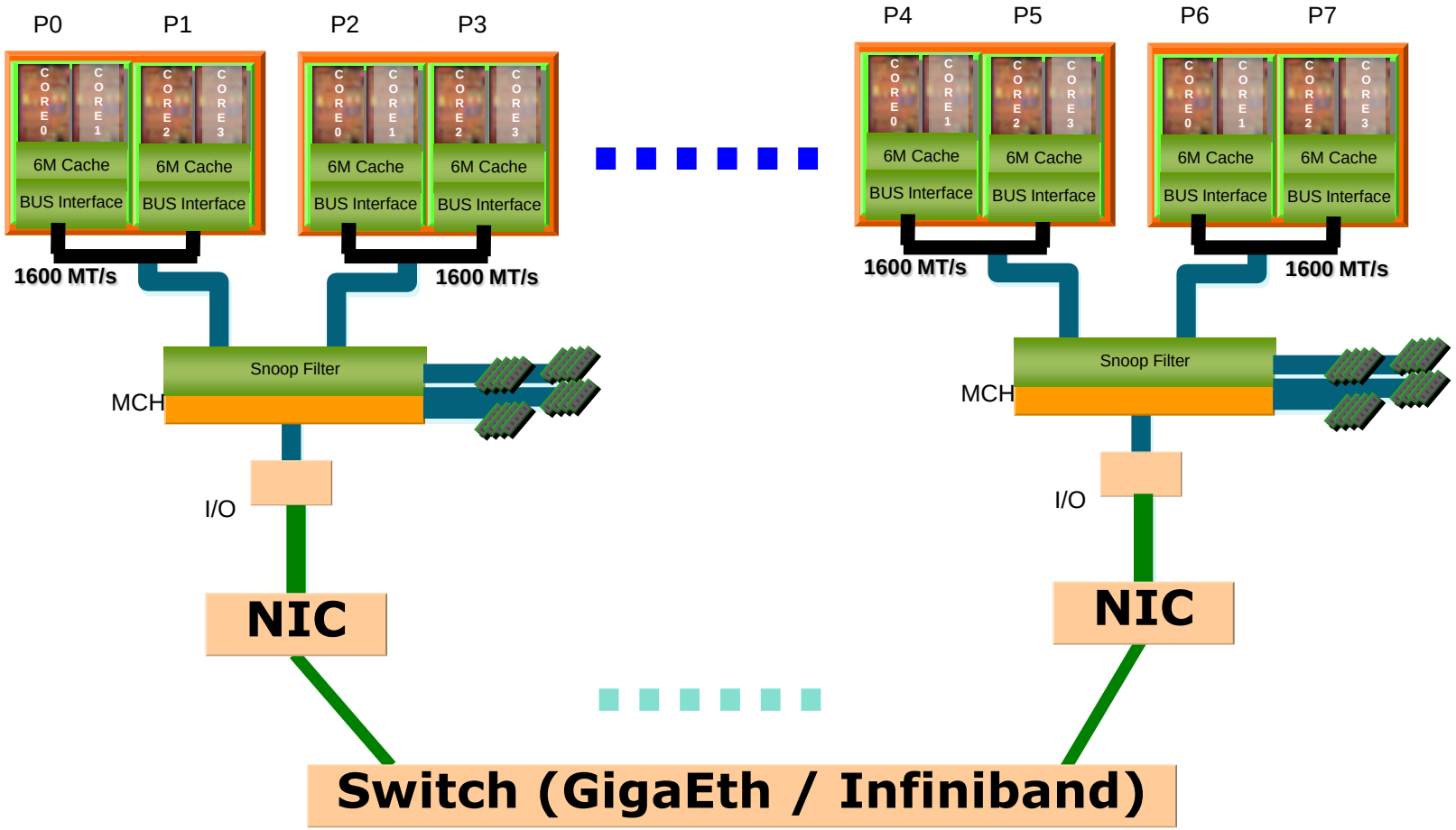
提高多线程程序的运行速度（并行调优）

- Intel Parallel Studio
  - Intel Inspector
  - Intel VTune Amplifier

分布内存并行程序运行模式/Node Scaling

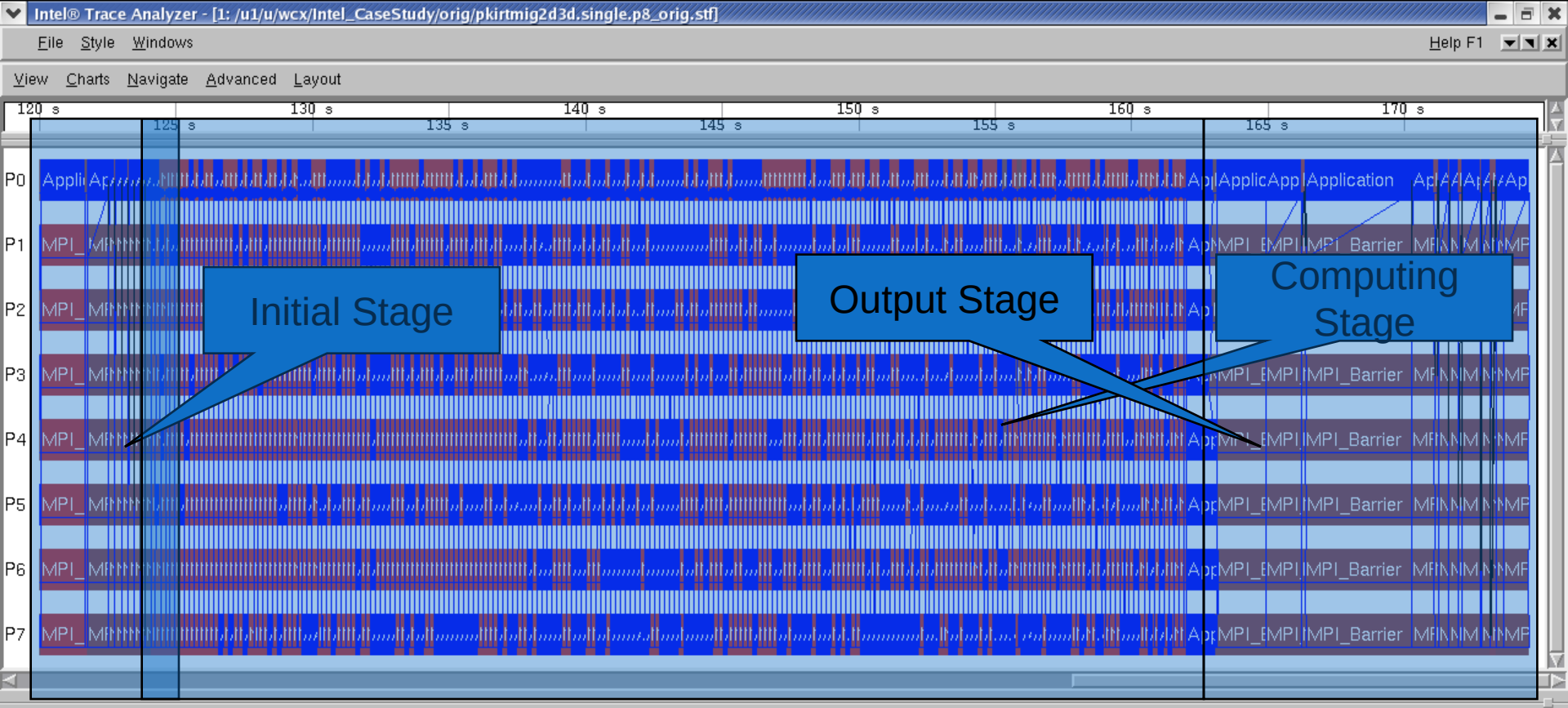
N个 双Socket四核服务器 可以有N*8个核

使用MPI分布式内存消息传递方式并行化已有的串行程序



如何提高并行程序的运行速度（并行调优）

- Paramon/Paratune
- Intel Parallel Studio
  - Intel MPI
  - Intel Trace Collector
  - Intel Trace Analyzer
  - Intel Cluster MKL
  - Intel MPI Benchmark



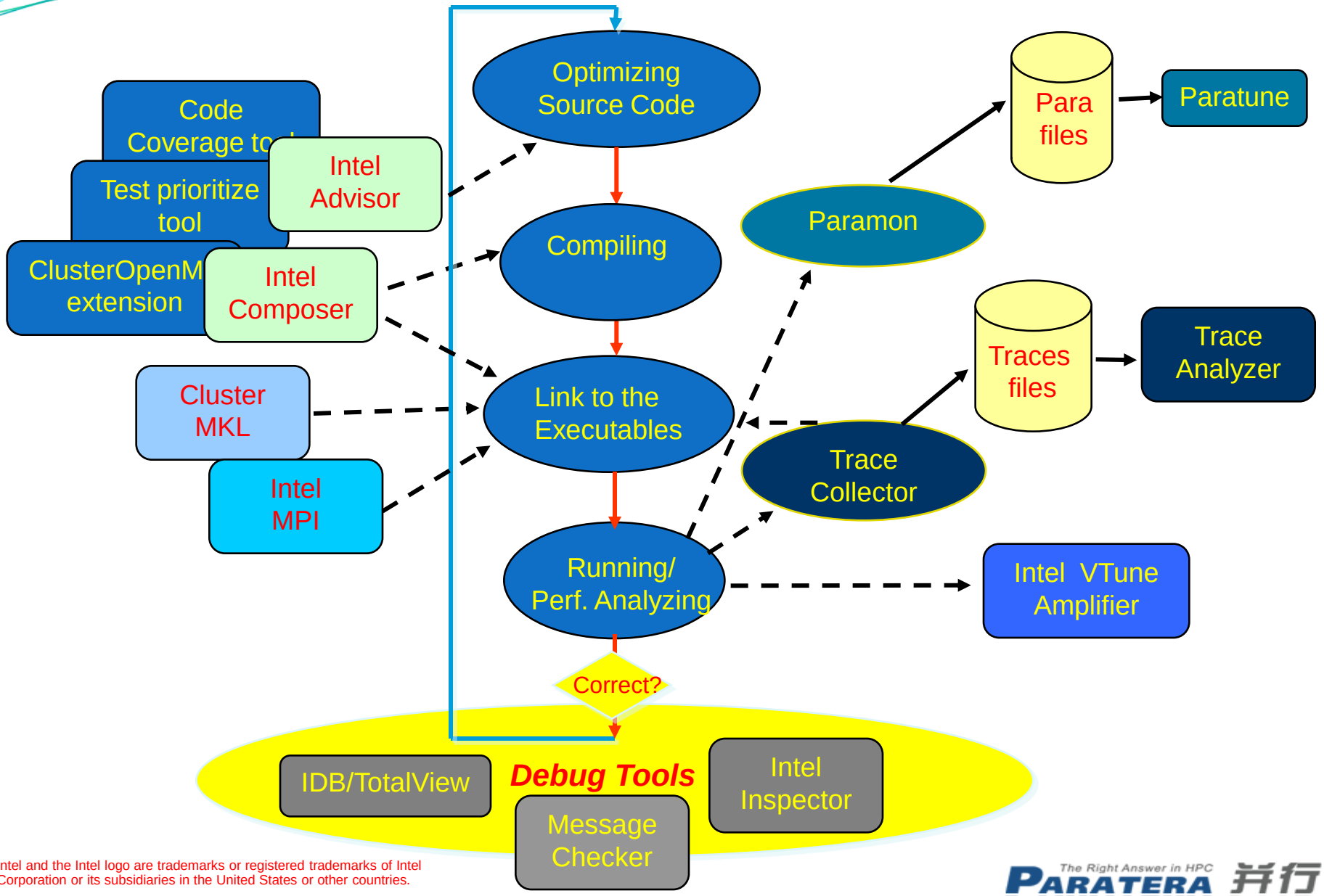
Flat Profile | Load Balance | Call Tree | Call Graph

Group All_Processes

Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
Group All_Processes					
Group Application	166.912 s		436.318 s	0	n.a.
MPI_Comm_size	0 s		0 s	8	0 s
MPI_Comm_rank	0 s		0 s	32	0 s
MPI_Finalize	0 s		0 s	8	0 s
MPI_Bcast	165.44 s		165.44 s	96248	1.7189e-3 s
MPI_Recv	880e-3 s		880e-3 s	33768	26.0602e-6 s
MPI_Send	880.003e-3 s		880.003e-3 s	33768	26.0603e-6 s
MPI_Barrier	102.206 s		102.206 s	889	114.967e-3 s

119.992 583, 174.534 666: 54.542 083 | sec. | All_Processes | MPI expanded in (Major Function Groups) | Tag Filter

使用英特尔工具集的性能导向的软件开发模型



总结

- 提高应用程序运行速度
- 改进算法（从用户视角）
- 软件调优
  - 串行（或并行程序中计算部分）优化
    - 提高代码效率 / 降低Cache Miss Rate / 提高向量化程度 / .....
    - 优化工具：Intel编译器 / Intel数学库 / 源代码优化(VTune) / .....
  - 并行（一般指并行程序中通信部分和IO部分）优化
    - Load Balance / Communication Pattern / .....
    - 并行优化工具：Intel线程工具 / Intel集群工具 / .....
- 并行化
  - 多线程（OpenMP）共享内存并行
  - 多进程（MPI）分布式内存并行