

IDF2013

英特尔信息技术峰会

动手实验室：利用英特尔® Parallel Studio XE 2013 将并行性从英特尔® 至强® 处理器扩展至英特 尔® 至强融核™ 协处理器

Shuo Li, Staff Application Engineer, Intel

Chao Yu, Technical Consulting Engineer, Intel

Jun Jin, Senior Application Engineer, Intel

SFTL002

议程

- 试验简介
- 什么是阶梯优化框架
- 第一步：以优化的工具和程序库作跳板
- 第二步：标量，单线程优化
- 第三步：向量化
- 第四步：并行化
- 第五步：从多核扩展到众核
- 总结

本课程演示文稿（**PDF**）发布在技术课程目录网站：
intel.com/go/idfsessionsBJ

该网址同时打印于会议指南中专题讲座日程页的上方

试验简介

欧式期权之蒙地卡罗方法

蒙地卡罗方法
从何而来



尼古拉斯 麦托坡里斯发明的
统计计算方法

蒙地卡罗在金融
的应用



费伦博尤 第一个把蒙地卡罗
带入定量金融界

- 简单的算法多次重复
- 利用中心极限定律

1. Sample a random path for S in a risk neutral world
2. Calculate the payoff from the derivative
3. Repeat steps 1 and 2 to get many sample values of the payoff from the derivative in a risk-neutral world
4. Calculate the mean of the sample payoff
5. Discount expected payoff at risk-free rate to get an estimate of the value of the option

最初的实现

- Use GCC 4.4.6
- C/C++ TR1 随机数产生程序
- 程序源文件清单

Driver.cpp

主程序文件

MonteCarlo.h

参数的定义

MonteCarloStepn.cpp

蒙地卡罗计算方法

Makefile

程序产生脚本

```
typedef std::tr1::mt19937      ENG;  // Mersenne Twister
typedef std::tr1::normal_distribution<float> DIST;
typedef std::tr1::variate_generator<ENG,DIST> GEN;

ENG  eng;
DIST dist(0,1);
GEN  gen(eng,dist);

for(int opt = 0; opt < OPT_N; opt++)
{
    float VBySqrtT = VOLATILITY * sqrt(T[opt]);
    float MuByT = (RISKFREE - 0.5 * VOLATILITY * VOLATILITY)
* T[opt];
    float Sval = S[opt];
    float Xval = X[opt];
    float val = 0.0, val2 = 0.0;

    for(int pos = 0; pos < RAND_N; pos++)
    {
        float callValue = max(0.0, Sval *exp(MuByT +
VBySqrtT *  gen()) - Xval);
        val  += callValue;
        val2 += callValue * callValue;
    }

    float exprt = exp(-RISKFREE *T[opt]);
    CallResult[opt] = exprt * val / (float)RAND_N;
    float stdDev = sqrt(((float)RAND_N * val2 - val * val)/
((float)RAND_N * (float)(RAND_N - 1)));
    CallConfidence[opt] = (float)(exprt * 1.96 * stdDev /
sqrtf((float)RAND_N));
}
```

你们的任务：让程序跑得更快更好

- 让程序在Intel® Xeon® processor 跑快，在Intel® Xeon Phi™ coprocessor上跑得更快
- 充分利用硬件资源
- 软件工具：Intel® Parallel Studio XE 2013 SP1
 - Intel® C/C++ Compiler
 - Intel® VTune™ Amplifier
 - Intel® Advisor XE
 - GDB Debugger
- 软件方法：逐步优化模式

让我们立即开始

什么是阶梯优化框架

阶梯优化框架

- 是一套能让程序员在多核和众核结构上表达程序并行性的方法和工具
- 终极目的：把没有优化的程序变成可扩展的，能在多核和众核上执行的高度并行程序



- 第一步：以优化的工具和程序库作跳板

- 第二步：标量，单线程优化

- 第三步：向量化

- 第四步：并行化

- 第五步：从多核扩展到众核

第一步：以优化的工具和程序库作跳板

用Intel® C/C++ Compiler产生蒙地卡罗欧式期权定价程序

- Intel® Compiler 完全与GCC兼容
- 你们桌上的手提电脑已经装上了Intel® Parallel Studio XE 2013
- 你可以用 `icpc -V` 来验证
- 运行环境变量脚本
`./opt/intel/composerxe/pkg_bin/compilervars.sh intel64`
- 再发一次make 命令这一次用CXX=icpc 作参数
`make CXX=icpc`
- 重新运行一次用Intel® C/C++ Composer XE产生的蒙地卡罗程序
`./MonteCarlo`
- 在发出的表格中记录下程序运行的性能

用 Intel® MKL 随机数产生器

- 加入Intel® MKL 头文件 `#include <mkl_vsl.h>`
- 定义一个缓冲区用来接受随机数
`float random[RAND_N];`
- 定义一个描述随机数系列的数据结构
`VSLSTREAMSTATEPTR Randomstream`
- 产生一个随机数系列并将其初始化。
`vslNewStream(&Randomstream, VSL_BRNG_MT19937, RANDSEE)`
- 用缓冲区来接受随机数系列
`vsRngGaussian (VSL_METHOD_SGAUSSIAN_ICDF,
Randomstream, RAND_N, random, 0.0, 1.0);`
- 在链接时加入 `-mkl` 重新产生和链接和启动
- 在发出的表格中记录下程序运行的性能

第二步：标量，单线程优化

优化程序的核心计算逻辑，减少不必要的重复

- 选定一个算法所要求准确度
 - Intel® Math Kernel Library (Intel® MKL) 准确模式 HA, LA, EP:
`vmlSetMode(VML_EP);`
 - 编译器的准确模式: `-fimf_precision=low,medium,high`
- 选定一个浮点数所要求的精度
 - 所有的常数都要给一个C/C++的 Type: `const float NUM = 1.0f`
 - 选用浮点精度一致的函数调用API `exp` for DP, `expf()` for SP
- 减小非正常浮点操作对性能的影响
 - 众核的非正常浮点运算成本更高 `-fp-modal fast=2`
`-fimf_domain_exclusion=15, 31`
- 为编译器产生优化的目标代码提供协助
 - 与常量有关的计算尽量提出循环语句
 - 编译时能确定的变量的计算尽量提前
 - 避免不必要的除法。除以一个常数能用乘这个数的倒数

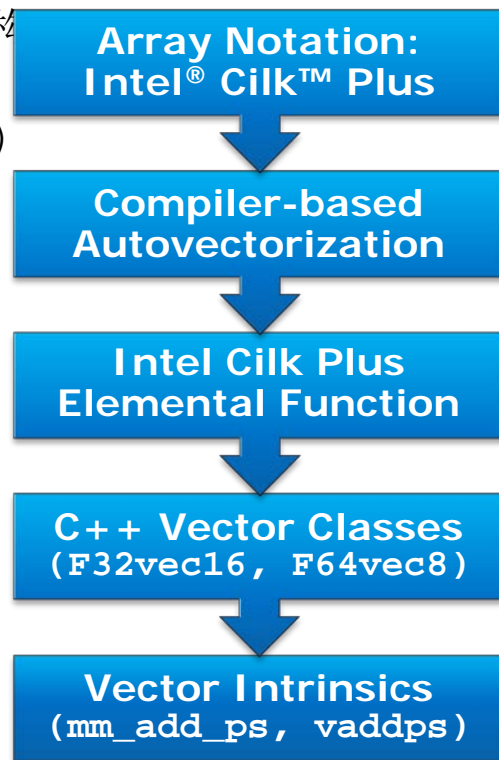
第二步：标量，单线程优化

- 察看给出的原程序代码，找出与C/C++语言有关的性能问题
 - 所有的常数要有一个C/C++ 的 Type
 - 明确地给出所有C/C++运行函数的数据类型
 - 把除以一个常数变成乘以这个数的倒数
- 找出你的算法所需要的准确度和浮点运算的精度
 - `-fimf-precision=low`
 - `-no-prec-div`
 - `-no-prec-sqrt`
- 排除与你的算法无关的浮点运算定义域
 - `-fimf-domain-exclusion=31`
- 其他的设置
 - `-fp-model fast=2`

第三步： 向量化

充分利用单指令多数据 (SIMD) 计算单元与向量处理单元 (VPU)

- 单指令多数据并行计算需要数据对齐
 - 将数据从结构式数组的存放格式 (AOS) 转化为数组式的结构的存放格式 (SOA)
 - 声明对齐的内存数据 `__attribute__((aligned(64))) float a;`
 - 动态分配对齐的内存数据: `_mm_malloc(size, align)`
 - 使用TBB:
`scalable_aligned_malloc(size, align)`
- 程序的分支会阻碍 SIMD指令执行
 - 一些条件分支语句可以用处理器的mask 指令改写
 - 函数调用可能会妨碍SIMD执行
- 首先尝试让编译器进行自动向量化
 - 提供编译器有关数据对齐, 数据依赖, 以及指针关联等信息
- 计算VPU 的使用率
 - `VPU_ELEMENTS_ACTIVE/VPU_INSTRUCTION_EXECUTED`
 - 双精度数据使用率<8 或 单精度数据使用率<16 时, 调查原因



提供编译器向量化的帮助信息

编译指示 (pragma)	语法(Semantics)
<code>#pragma ivdep</code>	编译器忽略不能确定的数据依赖。
<code>#pragma vector always [assert]</code>	循环能够向量化时，编译器忽略量化开销分析，总进行向量化。 循环不能向量化时，编译器通过"assert" 语句提供错误信息。
<code>#pragma novector</code>	提示编译器，即使在符合向量化条件下，也不进行循环向量化. 有时，向量化可能会带来性能开销，我们无需编译器进行向量化。
<code>#pragma vector aligned / unaligned</code>	向量化时，提示编译器使用对齐内存的操作指令（或不对齐内存的操作指令）。
<code>#pragma vector temporal / nontemporal</code>	在IA32与Intel 64的架构上，指示编译器使用 temporal/non-temporal 内存存取指令。多个变量时，使用逗号进行分隔。

使用simd 指示字强制循环向量化

– #pragma simd

- 语法: #pragma simd [<clause-list>]
 - 强制一个循环向量化的语法
 - 程序员: 需判断一个循环是否应该被向量化
 - 编译器: 强制将循环向量化或者给出错误信息

字句 (Clause)	语义 (Semantics)
none	强制将最内层循环进行向量化, 忽略数据间的依赖性。
vectorlength ($n_1[, n_2]...$)	指示向量化时, 向量长度(2, 4, 8, 16)。编译器编译时, 从中选择。
private ($var_1, var_2, ..., var_N$)	每次循环的私有标量。多个循环同时执行时, 各循环获得相同初始值。 变量的最终结果为最后执行循环的该变量值。
linear ($var_1:step_1, ..., var_N:step_N$)	声明变量的步长。步长值为正数, 且为向量长度的整数倍。
reduction ($operator: var_1, var_2, ..., var_N$)	声明reduction 变量。变量在循环结束时, 通过相应的reduction 操作累计最后的结果。
[no]assert	向量化失败时, 编译器通过assert 语句给出(或不给出)提示。 缺省为编译器给出向量化失败的提示。

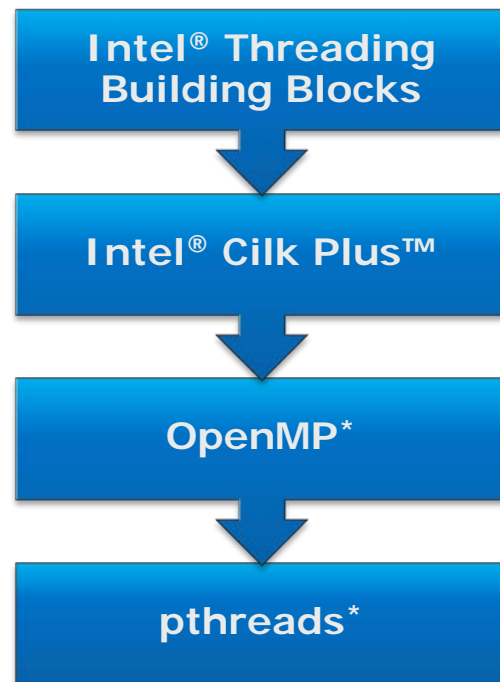
向量化蒙地卡罗欧式期权代码

- 找出需向量化的循环 - 其为最内层循环
- 改写动态内存分配， 使用对齐的内存分配语句
 - `Driver.cpp malloc(int size) -> _mm_malloc(size, align)`
 - `Driver.cpp free() -> _mm_free()`
- 将max 宏改为inline 代码
 - `float callValue=max(0.0,Sval*expf(MuByT+VBySqrtT*random[pos])-Xval);`
改为:
 - `float callValue=Sval*expf(MuByT+VBySqrtT*random[pos])-Xval;`
 - `callValue = (callValue > 0) ? callValue : 0`
- 增加向量化指示语句
 - `#pragma vector aligned`
 - `#pragma simd reduction(+:val) reduction(+:val2)`
 - `#pragma unroll(4)`
- 改写Makefile
 - 在编译器编译选项行，增加 `-xAVX -vec-report6` 编译选项

第四步：并行化

充分利用处理器的核与线程

- 在上层代码分配计算工作
 - 使用粗粒度的并行
- 控制好线程创建的开销
- 减少多线程同步的开销
- 注意多线程的负载均衡
 - 考虑任务分配方法：静态还是动态



并行化的方法 – OpenMP*

- C/C++ 的 OpenMP* 语法:

`#pragma omp construct [clause [clause]...]`

- 多种并行语法的支持:

- 并行任务与代码段
- 并行循环
- 同步语法
 - 临界区 (critical sections)
 - 同步分界 (barriers)
- 原子操作与顺序操作
- 并行代码中的顺序执行区域

- 支持 offloading 程序的扩展语法 – OpenMP 4.0 RC2

- 使用英特尔 Offload 的扩展语法 (Language Extensions for Offload)

`#pragma omp target or #pragma offload`

两种语法都可以, 没有性能区别。

```
#pragma omp parallel sections
{
    #pragma omp section
    {
        BinomialTemplate<F32vec8, float>(callResult, S,
                                          X, T, R, V, N, NUM_STEPS);
    }
    #pragma omp section
    #pragma offload target(mic:0) in(S, X, T:length(N))
        out(MICEExpected, MICConfidence:length(N))
    {
        montecarlo(S, X, T, MICEExpected, MICConfidence);
    }
}
```

并行化的方法 - 共有任务的 OpenMP*例子

串行代码

```
for (i = 0; i < N; i++) a[i] = a[i] + b[i];
```

使用OpenMP*
parallel region 的例子

```
#pragma omp parallel
{
    int id, i, Nthrds, istart, iend;

    id = omp_get_thread_num();
    Nthrds = omp_get_num_threads();
    istart = id * N / Nthrds;
    iend = (id + 1) * N / Nthrds;

    for (i = istart; i < iend; i++)
        a[i] = a[i] + b[i];
}
```

使用 OpenMP*
Worksharing construct
的例子

```
#pragma omp parallel
#pragma omp for
    for (i = 0; i < N; i++) a[i] = a[i] + b[i];
```

第四步：并行化

- 在外层循环，加入

```
#pragma omp parallel for
```

- 在C/C++编译器选项中，添加 `-openmp`
- 重新运行程序

```
./MonteCarlo
```

- 设置环境变量，重新运行程序

```
export KMP_AFFINITY="compact,granularity=fine"
```


第五步： 从多核扩展到众核

把向量化的、并行的代码扩展到更多线程

- 目标平台上特定的优化手段
 - 利用EMU中的性能卓越的超然函数
- 线程绑定
 - 生成数目是党的工作线程
 - 选择想要的线程邦定类型
- 数据切块（Data Blocking）
 - 提升内存访问效率
 - 缓解内存带宽的压力
 - 强化数据的多次使用
- 其他优化手段
 - 改善数据局部性
 - 尽可能减少内存访问

安装Intel® Xeon Phi™协处理器的平台

- 采用Intel® Xeon Phi™ 的Intel® Xeon® E5 2670 平台
- 确认能够调用Intel® C/C++ 编译器
 - `icc -V`
- 编译 Intel Xeon Phi协处理器上的Native代码
 - 只需要功过修改Makefile （把-xAVX替换成 `-mmic` ）

```
source /opt/intel/composerxe/pkg_bin/compilervars.sh intel64
```

```
export KMP_AFFINITY="compact,granularity=fine"
```

第五步：超越函数

- 用EMU中性能卓越的超越函数
- 内层循环调用 `expf(MuByT + VBySqrtT * random[pos]);`
- 改成调用`exp2f` 同时通过系数`M_LOG2E`调整参数
- 结合对`MuByT` 和`VBySqrtT`的乘法

```
float VBySqrtT = VLOG2E * sqrtf(T[opt]);  
float MuByT = RVVLOG2E * T[opt];
```

第五步：线程绑定

- 包含头文件 `#include <omp.h>`
- 在代码开始的时候插入下列代码

```
#ifndef _OPENMP
kmp_set_defaults("KMP_AFFINITY=compact,granularity=fine");
#endif
```

- Makefile中增加选项 `-opt-threads-per-core=4`

第五步：数据切块

- 一维数据的数据切块.

- L2: 每核521K, 每线程128K
- 应用程序可以使用的部分是64KB 16K 单精浮点
- BLOCKSIZE = 16*1024

```
const int nblocks = RAND_N/BLOCKSIZE;
for(block=0; block<nblocks; ++block){
    vsRngGaussian(VSL_METHOD_SGAUSSIAN_ICDF,
Randomstream, BLOCKSIZE, random, 0.0f,
1.0f);
```

<<<Existing Code>>>

```
// Save intermediate result here
```

```
h_CallResult[opt] += val;
```

```
h_CallConfidence[opt] += val2;
```

```
}
```

- 改变内层循环 (RAND_N -> BLOCKSIZE)

- 把期权定价数据的计算循环移动到最外层循环

```
#pragma omp parallel for
for(int opt = 0; opt < OPT_N; opt++)
{
    const float val=h_CallResult[opt];
    const float val2=h_CallConfidence[opt];
    const float expprt=exp2f(-RLOG2E*T[opt]);
    h_CallResult[opt]= expprt*val*INV_RAND_N;
    const float stdDev= sqrtf((F_RAND_N * val2 -
val * val) * STDDEV_DENOM);
    h_CallConfidence[opt]= (float)(expprt * stdDev *
CONFIDENCE_DENOM);
```

- 在三重循环外面增加初始化循环

```
#pragma omp parallel for
for(int opt = 0; opt < OPT_N; opt++)
{
    h_CallResult[opt] = 0.0f;
    h_CallConfidence[opt] = 0.0f;
}
```

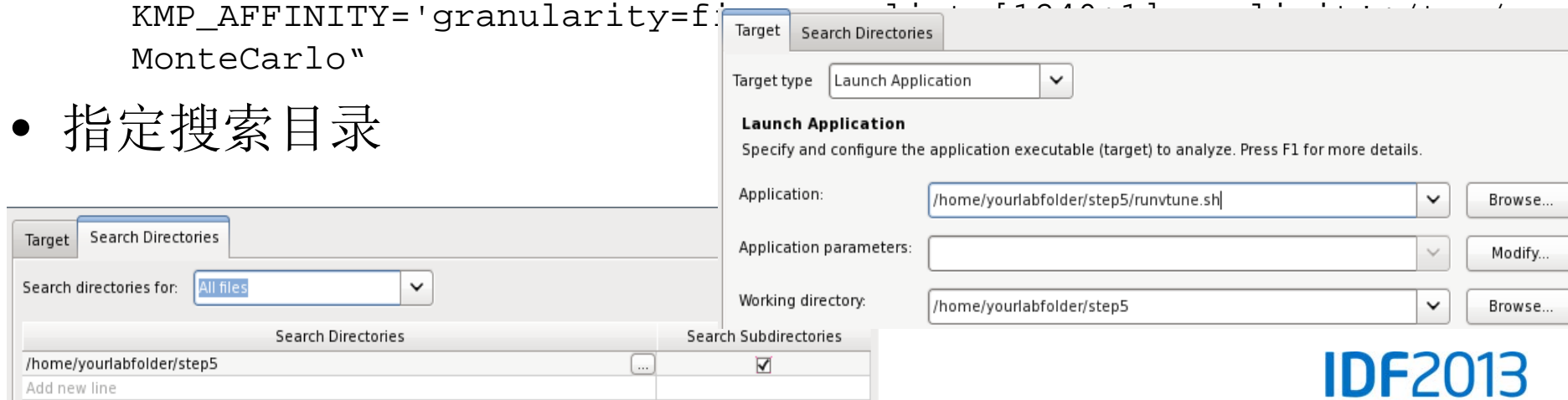
在Intel® Xeon Phi™上运行程序

- 在Intel® Xeon Phi™编译Native程序
 - 只需要调整Makefile，把原来的`-xAVX`改成`-mmic`
- 把执行程序从Host拷贝到协处理器上
 - 通过命令行拷贝：`scp MonteCarlo mic0:`
 - 登录到协处理器的执行环境 `ssh mic0`
 - 设置环境变量 `%export LD_LIBRARY_PATH=.`
 - 运行程序 `./MonteCarlo`
 - 在表格中记录性能数据

采用Intel® VTune™ Amplifier 性能分析工具的性能调优方法

采用Intel® VTune™ Amplifier XE工具分析Intel® Xeon Phi™协处理器上的Native代码

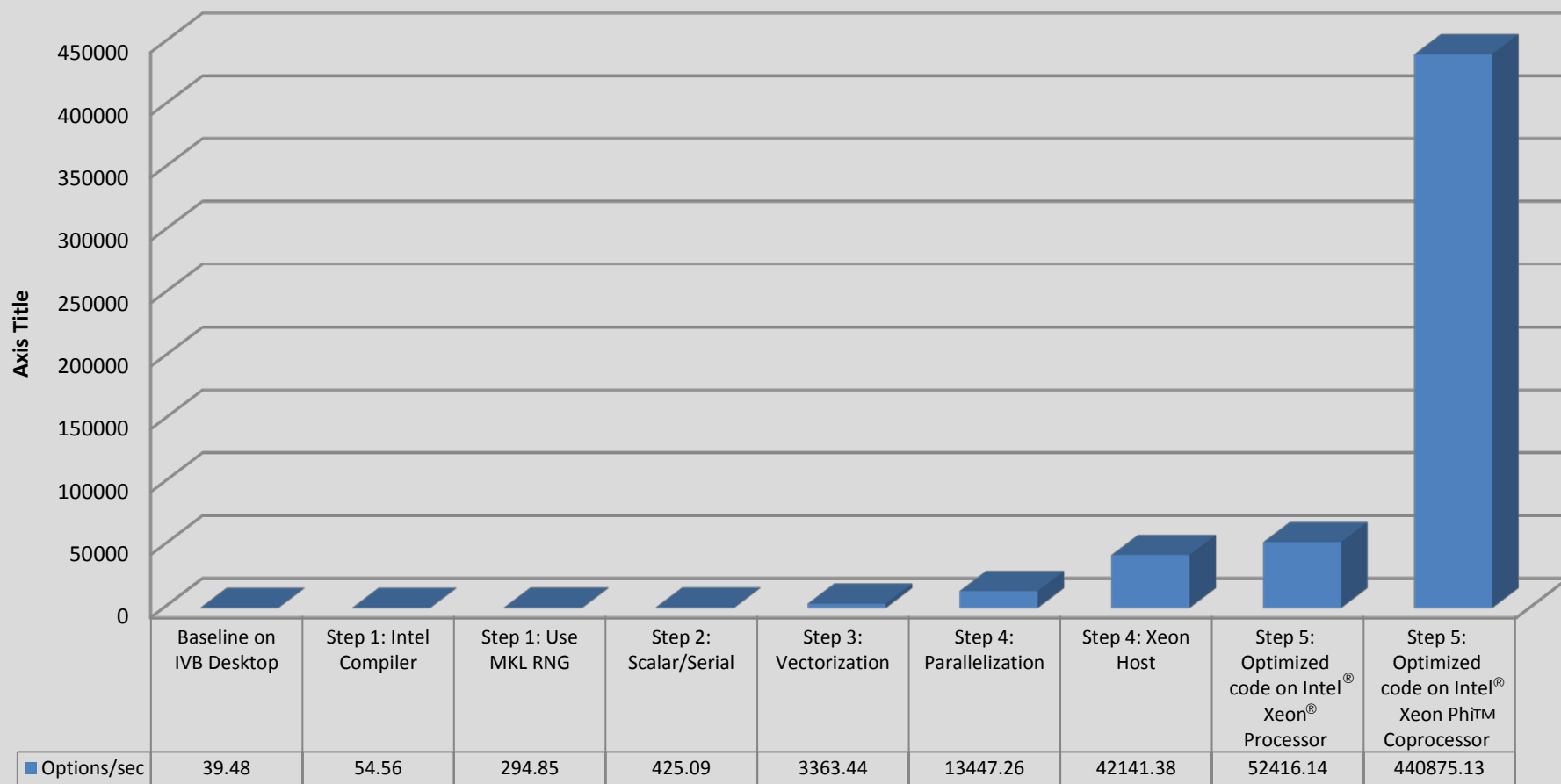
- 命令行启动Intel® Vtune™ Amplifier: `amplxe-gui`
 - 必须先source
`/opt/intel/composerxe/vtune_amplifier_XE_2013`
- 创建新的项目 File>New>Project
- 指定分析Xeon Phi协处理器上的代码
 - 通过脚本runvtune.sh 启动程序
 - `ssh mic0 "export LD_LIBRARY_PATH=/tmp;export OMP_NUM_THREADS=240;export KMP_AFFINITY='granularity=fine;compact,1,0';/opt/intel/composerxe/bin/intel64/libcompilershared/libcrt_ia32.so.1.0.0.1;./runvtune.sh"`
 - `MonteCarlo"`
- 指定搜索目录



总结

总结

Monte Carlo European Option Pricing (Single Precision, Path length = 256k)



行动与号召

- 采用Intel® Xeon Phi™协处理器运行高度并行的应用程序
- 通过Intel® Parallel Studio XE 2013更好的实现并行性
- 通过阶梯式的优化框架，把并行程序从Intel® Xeon® Processor有效的扩展到Intel® Xeon Phi™协处理器上

Legal Disclaimer

Do not translate

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

- A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.
- Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.
- The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.
- Intel product plans in this presentation do not constitute Intel plan of record product roadmaps. Please contact your Intel representative to obtain Intel's current plan of record product roadmaps.
- Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. Go to: http://www.intel.com/products/processor_number.
- Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.
- Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: <http://www.intel.com/design/literature.htm>
- Intel, Xeon, Xeon Phi, VTune, Sponsors of Tomorrow and the Intel logo are trademarks of Intel Corporation in the United States and other countries.
- *Other names and brands may be claimed as the property of others.
- Copyright ©2013 Intel Corporation.

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Legal Disclaimer

Do not translate

Pick the ones needed - remove others

- Software Source Code Disclaimer: Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.
- Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Risk Factors

Do not translate

The above statements and any others in this document that refer to plans and expectations for the first quarter, the year and the future are forward-looking statements that involve a number of risks and uncertainties. Words such as “anticipates,” “expects,” “intends,” “plans,” “believes,” “seeks,” “estimates,” “may,” “will,” “should” and their variations identify forward-looking statements. Statements that refer to or are based on projections, uncertain events or assumptions also identify forward-looking statements. Many factors could affect Intel’s actual results, and variances from Intel’s current expectations regarding such factors could cause actual results to differ materially from those expressed in these forward-looking statements. Intel presently considers the following to be the important factors that could cause actual results to differ materially from the company’s expectations. Demand could be different from Intel’s expectations due to factors including changes in business and economic conditions; customer acceptance of Intel’s and competitors’ products; supply constraints and other disruptions affecting customers; changes in customer order patterns including order cancellations; and changes in the level of inventory at customers. Uncertainty in global economic and financial conditions poses a risk that consumers and businesses may defer purchases in response to negative financial events, which could negatively affect product demand and other related matters. Intel operates in intensely competitive industries that are characterized by a high percentage of costs that are fixed or difficult to reduce in the short term and product demand that is highly variable and difficult to forecast. Revenue and the gross margin percentage are affected by the timing of Intel product introductions and the demand for and market acceptance of Intel’s products; actions taken by Intel’s competitors, including product offerings and introductions, marketing programs and pricing pressures and Intel’s response to such actions; and Intel’s ability to respond quickly to technological developments and to incorporate new features into its products. The gross margin percentage could vary significantly from expectations based on capacity utilization; variations in inventory valuation, including variations related to the timing of qualifying products for sale; changes in revenue levels; segment product mix; the timing and execution of the manufacturing ramp and associated costs; start-up costs; excess or obsolete inventory; changes in unit costs; defects or disruptions in the supply of materials or resources; product manufacturing quality/yields; and impairments of long-lived assets, including manufacturing, assembly/test and intangible assets. Intel’s results could be affected by adverse economic, social, political and physical/infrastructure conditions in countries where Intel, its customers or its suppliers operate, including military conflict and other security risks, natural disasters, infrastructure disruptions, health concerns and fluctuations in currency exchange rates. Expenses, particularly certain marketing and compensation expenses, as well as restructuring and asset impairment charges, vary depending on the level of demand for Intel’s products and the level of revenue and profits. Intel’s results could be affected by the timing of closing of acquisitions and divestitures. Intel’s current chief executive officer plans to retire in May 2013 and the Board of Directors is working to choose a successor. The succession and transition process may have a direct and/or indirect effect on the business and operations of the company. In connection with the appointment of the new CEO, the company will seek to retain our executive management team (some of whom are being considered for the CEO position), and keep employees focused on achieving the company’s strategic goals and objectives. Intel’s results could be affected by adverse effects associated with product defects and errata (deviations from published specifications), and by litigation or regulatory matters involving intellectual property, stockholder, consumer, antitrust, disclosure and other issues, such as the litigation and regulatory matters described in Intel’s SEC reports. An unfavorable ruling could include monetary damages or an injunction prohibiting Intel from manufacturing or selling one or more products, precluding particular business practices, impacting Intel’s ability to design its products, or requiring other remedies such as compulsory licensing of intellectual property. A detailed discussion of these and other factors that could affect Intel’s results is included in Intel’s SEC filings, including the company’s most recent Form 10-Q, report on Form 10-K and earnings release.

Rev. 1/17/13