Com4521 / Com6521：使用GPU进行并行计算分配：第2部分

截止日期：2019年5月13日星期一17:00（第12周）

印记 最后编辑：09/03/2018

作业2（共2个）占总作业标记的70%。总分配标记（第1部分和第2部分）占模块总标记的80%。

对于代码功能和性能，赋值2标记的权重为50%，通过书面报告表示理解的权重为50%。

文件变更

这是作业文件的第1版。此处将记录对本文档的任何更正或更改，并将更新发送至当然谷歌组邮件列表。

介绍

任务的目的是测试您在GPU上实现高效代码的理解和技术能力。您需要对基准和优化简单图像处理问题的实现进行基准测试。您已在Assignment 1中实现了串行和多核版本，您需要实现相同任务的GPU版本。这项任务的重点在于您逐步改进代码以融合高效实现的能力。为了证明这一点，您需要记录（在书面报告中）为改善性能（并确保正确性）而进行的任何设计考虑。例如，您应该使用基准测试来演示实现的更改如何导致性能改进，或者通过理论解释为什么选择特定方法。除非您还在书面报告中表明了对如何逐步完善实施以达到最终解决方案的理解，否则只需提交一段性能优异的代码就不会在评估中获得高分。

作业要求（代码）

您需要在GPU上实现像素过滤器（使用Assignment 1文档中的相同问题定义）。您应该使用您的切换1作为您应该已经执行IO和CPU / OpenMP模式的起始代码。您的作业1代码在程序模式为CUDA时已经有占位符。为了将GPU代码添加到您的作业1切入，您需要将现有C文件重命名为CUDA扩展名（*.cu），然后再将其导入新的Visual Studio，NVIDIA CUDA项目。您的程序应该接受额外的CUDA模式参数，您应该更新print_help（）函数以反映新的CUDA选项。您应该使用作业1的参考反馈，以确保您的GPU代码高效且正常工作。如果您之前未正确实现此操作，则可能需要更新文件IO代码（否则您的GPU代码可能会产生错误的结果）。您仍然可以使用mode program参数运行CPU和OpenMP代码，但不会重新评估此代码

作为任务的一部分2。

定时：

GPU版本代码的程序参数与先前的赋值相同。您应该注意确保使用适当的技术准确计算GPU代码的时间。

并行实施：

有两种明显的方法可以在GPU上实现像素化技术。您可以并行化每个马赛克单元格，也可以并行化每个输入像素。每种方法可能更适合某些尺寸的马赛克单元格大小C.您可以实现任一方法（或两者的混合），但是，您必须证明您在文档中的决定是正确的。您可以考虑实施这两种技术，以便在考虑不同的C值时提供哪种方法有利的证据。您应该考虑一系列输入图像大小，以确保在任一版本中您都有足够的线程来充分利用该设备。较大的输入图像可能较慢，但可能会提供更好的设备利用率。

文件要求

您需要记录代码的实现。更具体地说，您需要比较和对比各种实现技术，以展示您如何融合特定实现。特别是，您应该在发布模式下对代码进行基准测试，以比较替代技术（例如在适当的情况下使用各种GPU内存缓存），并解释为什么一种实现技术（或您所做的优化）优于另一种技术. .一些有趣的基准或讨论的例子包括：

- 并行化方法？
- 在GPU内存中布局或表示数据的不同方法，以确保合并的访问模式。例如结构阵列与阵列结构。
- 使用各种GPU内存缓存（纹理/只读，常量，共享内存）来减少全局内存读取的数量？注意：并非所有问题都适用于您的问题，但应该讨论原因。
- 您为提高性能而进行的任何GPU优化？通过基准测试或分析对性能进行调查的描述。
- 您已应用于代码的GPU版本的实现或优化技术的任何其他有趣方面。

基准测试应始终在Visual Studio中的“发布”模式下完成，并且计划的单次运行的计时结果将通过多个独立程序运行进行平均。基准测试应考虑图像大小和马赛克过滤器大小C（在第1部分文档中定义）的各种值，以演示性能缩放。对于代码的每个重大改进，尝试在更改之前和之后显示代码的性能。您应该突出（使用短代码示例）您所做的任何新颖方面或优化。

项目交接

您应该通过MOLE将您的程序代码与单个zip文件中的单个pdf一起提交。您还应该包括Visual Studio解决方案和任何项目文件。你的代码应该构建

在发布模式配置中，Diamond计算机机房4（实验室）计算机上没有错误或警告（智能感知除外）。如果您尚未完成整个作业，则应提交您所做的任何事情。您的代码不应依赖任何第三方库或工具（CUDA或OpenMP附带的库除外）。

印记

转让第2部分的标记将分发如下：

- 50%的任务是针对编码方面的。这个百分比的一半用于编程和优化的质量（包括代码的性能），另一半用于满足要求。
- 50%的任务是用于生成描述您为实现和优化代码而采取的流程的文档。这应该包括文档要求中描述的方法的基准测试和迭代改进。

在评估您的工作时，将考虑代码方面的以下要求。

1. GPU代码在功能上是否正确？即技术是否已正确实施并产生正确的结果？将使用许多测试用例来针对参考实现对此进行评估。
2. 您是否设法适当地使用GPU，确保设备具有足够的并行度以适应所有马赛克单元尺寸？
3. 代码的迭代改进是否产生了足够优化的最终GPU程序？
4. 代码是否充分利用了内存带宽？
5. GPU代码在降低价值时是否会避免竞争条件？
6. GPU代码是否使用有效的缓存来减少全局内存负载的数量？
7. 是否有任何编译器警告或危险的内存访问（超出分配的内存范围）？你的程序是否释放任何分配的内存？
8. 您的代码是否结构清晰且评论良好？

在评估您的文档时，将考虑以下内容，并作为讨论代码增量改进的指南。

1. 技术描述及其实现方式。选择并行化方法是一个很好的理由吗？
2. 是否已对使用良好的内存访问模式和适当的缓存技术进行了适当的调查？是否对基准测试结果给出了很好的解释？
3. 您的文档是否描述了对代码的优化并显示了这些优化的影响？
4. 是否有关于所有三个版本代码之间性能差异的基准测试和讨论？

开发代码和文档的技巧

如果您无法在GPU上实现该技术的所有方面，那么您应该默认使用该部分的CPU或OpenMP版本，以便您的代码构建并执行生成正确的结果。同样，如果您应用的技术不能提高性能，则应将其包含在文档中，并解释您对其未能按预期工作的原因的信念/理解。您可以使用#define来允许您的代码构建在不同的版本中

比较技术更直接的前进。

您应该评论您的代码，以明确您所做的事情。您应该测试您的代码，以确保它适用于所有图像大小，values值和图像输入文件。对于不正确的值和输入文件（例如*c*= -10或具有不正确像素值数的输入文件），您的代码应优雅地退出并显示有用的错误消息。您的代码永远不应该读取或写入分配的内存。

作业帮助

一般实验课程将提供您的作业帮助。对于实验室之外的具体问题，您应该使用Google讨论组课程。