周报-向嘉豪 (2025 年 3 月 17 日)

摘要：本周对 SPHINCS+ 签名算法进行了动态并行优化研究，重点探索了自适应线程分配（Adaptive Thread）技术。通过理论分析提出了线程性能模型 $T(g_i, t) = \alpha_i + \frac{\beta_i}{t} + \gamma_i \cdot t$，并导出最优线程数计算公式 $t_i^* = \sqrt{\frac{\beta_i}{\gamma_i}}$。实验验证表明，对于密钥生成操作，最优线程配置 (128×256) 相比默认配置 (512×32) 提升了 22.7% 的性能。同时完成了论文的绪论和自适应线程分配部分的修改，优化了论文结构和学术论证。

下周计划：1）将自适应线程研究扩展到签名和验证操作；2) 设计并实现自动化框架，用于确定各类密码算法操作的最佳线程配置。

1 实验验证

1.1 Adaptive Thread

本周我们针对 SPHINCS+ 签名算法进行了动态并行优化研究，重点探索了自适应线程分配（Adaptive Thread）技术在提升吞吐量方面的效果。

1.1.1 理论分析

传统的 CUDA 并行实现通常将线程数量设为可用的最大值，但我们的研究表明这并非最优方案。对于每个内核函数 g_i，存在一个最佳的线程数量 t_i，可以使其性能达到最大化。我们提出了自适应线程函数 $AT : G \to T$，它将每个函数 $g_i \in G$ 映射到其最优线程数 $t_i \in T$。

为准确定义此映射，我们通过经验性能建模方法进行研究：

$$T(g_i, t) = \alpha_i + \frac{\beta_i}{t} + \gamma_i \cdot t \tag{1}$$

其中：α_i 表示固定开销，$\frac{\beta_i}{t}$ 表示并行加速部分，$\gamma_i \cdot t$ 表示线程管理开销，而 t 是线程数量。

通过微分计算，可以得到函数 g_i 的最优线程数为：

$$t_i^* = \sqrt{\frac{\beta_i}{\gamma_i}} \tag{2}$$

1.1.2 实验结果

我们采用 CUSPX 实现对 SPHINCS+ 算法进行了不同线程配置的性能测试，重点关注密钥生成操作的执行效率。表1展示了不同 thread-block 配置下的性能对比。

表 1: 不同线程配置下 SPHINCS+ 密钥生成性能对比

Block×Thread 配置	每操作耗时 (ms)	相对默认配置性能提升
默认配置 (512×32) [WDC+25]	0.0022	基准
128×256	0.0017	22.7%
128×64	0.0025	-13.6%
128×512	0.0018	18.2%

实验结果表明，对于 SPHINCS+ 的密钥生成操作，128×256 的线程配置比默认的 512×32 配置提供了 22.7% 的性能提升。这验证了我们的理论假设，即并非所有操作都应使用最大线程数。特别是，在 RTX-4090 这样的硬件上，尽管理论上有 128 个 SM×1024 线程可用，但并非所有计算类型都能从最大化线程数中获益。

1.1.3 实施策略

基于上述研究，我们提出了以下实施策略：首先对每个关键函数进行离线性能剖析，精确量化不同线程配置下的执行效率；然后构建函数到最优线程数的映射表，为常见操作预先确定理想配置；最后实现动态线程分配机制，在运行时根据具体计算负载、硬件状态和并行度需求自动调整线程参数，从而在各种操作场景下实现接近最优的性能表现。

接下来，我们将扩展这项研究到签名和验证操作，并开发一个自动化框架来确定各种密码操作的最佳线程配置。

2 论文写作

本周完成了论文的两个关键部分的修改和完善：

1. **绪论部分**：精简了 PQC 背景介绍，突出 SPHINCS+ 与 SLH-DSA 的关系；重构了相关工作部分，明确指出当前 GPU 实现的两个主要效率瓶颈：统一的最大线程分配策略和次优的线程性能；强化了论文贡献点的表述，突出自适应线程分配方法的创新性。

2. **自适应线程分配部分**：构建了基于函数特性的性能建模方法，采用 $T(g_i, t) = \alpha_i + \frac{\beta_i}{t} + \gamma_i \cdot t$ 模型来平衡并行加速与线程管理开销；提出最优线程数计算公式 $t_i^* = \sqrt{\frac{\beta_i}{\gamma_i}}$；设计了动态实现算法，通过离线分析和运行时调整实现资源的最优利用。

参考文献

[WDC+25] Ziheng Wang, Xiaoshe Dong, Heng Chen, Yan Kang, and Qiang Wang. Cuspx: Efficient gpu implementations of post-quantum signature sphincs⁺. *IEEE Transactions on Computers*, 74(1):15–28, 2025.