Linux 内核性能测试框架的实现与优化

中期报告

杨 扬

指导教师：王生原　陈渝

清华大学计算机科学与技术系

2013 年 4 月 22 日

工作计划—时间表

时间段	工作内容
1–4 周	开题调研及初步设计
5–6 周	完成系统监视器及相关数据提取
7–8 周	实现数据流格式化及分析
9–12 周	实现并调试 10-bisect 算法
13–15 周	完成最后的系统调试和优化
16 周及以后	撰写毕业设计论文

目前进度

经过几周的努力，目前已经完成的进度是：

1. 已完成全部系统监视器的编写
2. 已重构并完成全部系统监视器输出的分析脚本
3. 已经能够在 KVM 虚拟机中自动完成整个测试流程
4. 进行数据的比较分析及可视化比较
5. 实现了简单的 Performance Regression 判断算法

系统监视器

目前，所有的系统监视器和对应的数据提取和格式化代码都已经完成。
其中，我们的系统监视器包括覆盖三个方面：

- CPU 类：interrupts, sched_debug, softirqs, vmstat, latency_stats, lock_stats, proc-vmstat
- 内存类：buddyinfo, meminfo, slabinfo, numa-meminfo, pagetypeinfo, numa-vmstat, numa-numastat
- I/O 类：mountstats, nfsstat, iostat

监视器实现及数据解析

监视器的实现

1. 直接使用现成的程序输出
2. 读取/proc 文件夹下面的相关系统状态文件

监视器数据解析

1. 阅读现成监控程序的帮助
2. 阅读 Linux 中与/proc 相关的输出代码
3. 解析之后输出成为 yaml 格式方便处理

数据汇总及处理

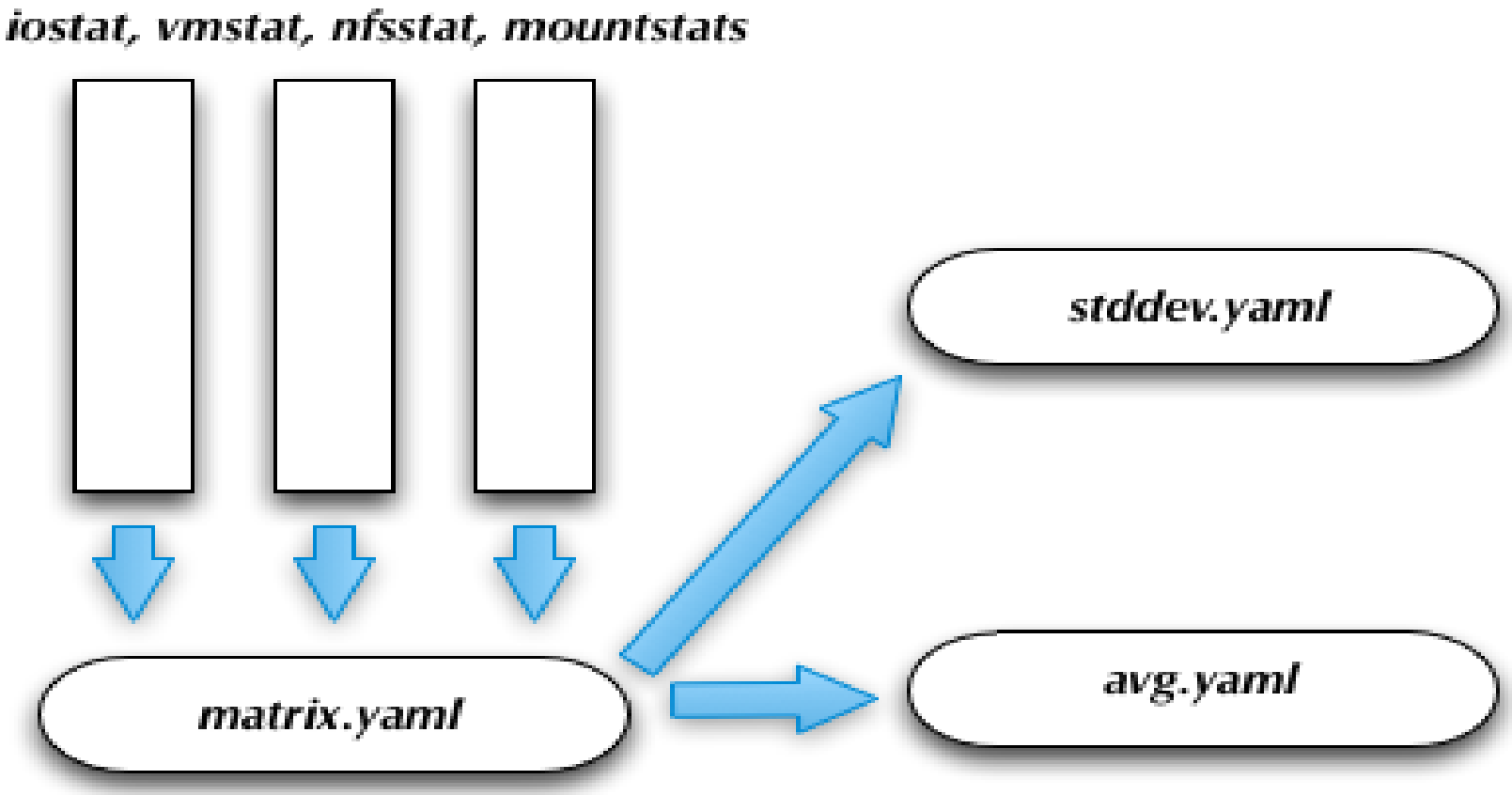


Figure : 汇总及处理

数据比较分析

已经可以进行的数据的比较和分析：

1. 比较多个 commit
2. 比较多个 config
3. 比较同一个 config,commit 的多次测试
4. 计算变化（包括上升和下降）的比例

下面我们以 vmstat 和 iostat 这两个监视器中的两项指标进行比较分析（见下页）

数据比较分析（续）

1		2		3	
3.00	+0.0%	3.00		0	kvm_lkp/micro/dd-wri
18.95	+0.8%	19.10	-0.4%	18.88	kvm_lkp/micro/dd-wri
3.00	+33.3%	4.00		0	kvm_lkp/micro/dd-wri
24.95	+4.6%	26.10	-24.3%	18.88	TOTAL vmstat.cpu.sy

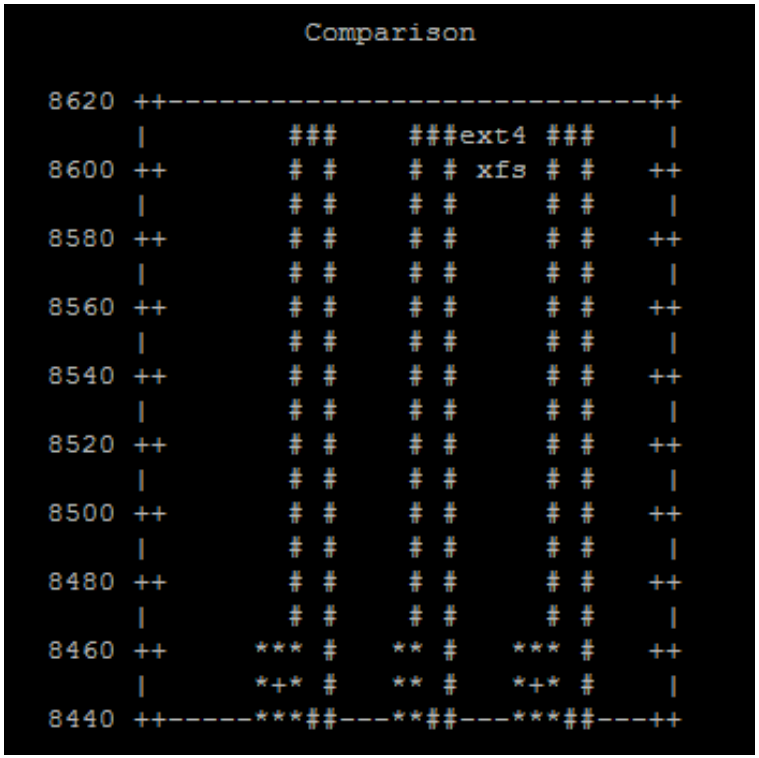
Figure : 比较分析 vmstat.cpu.sy

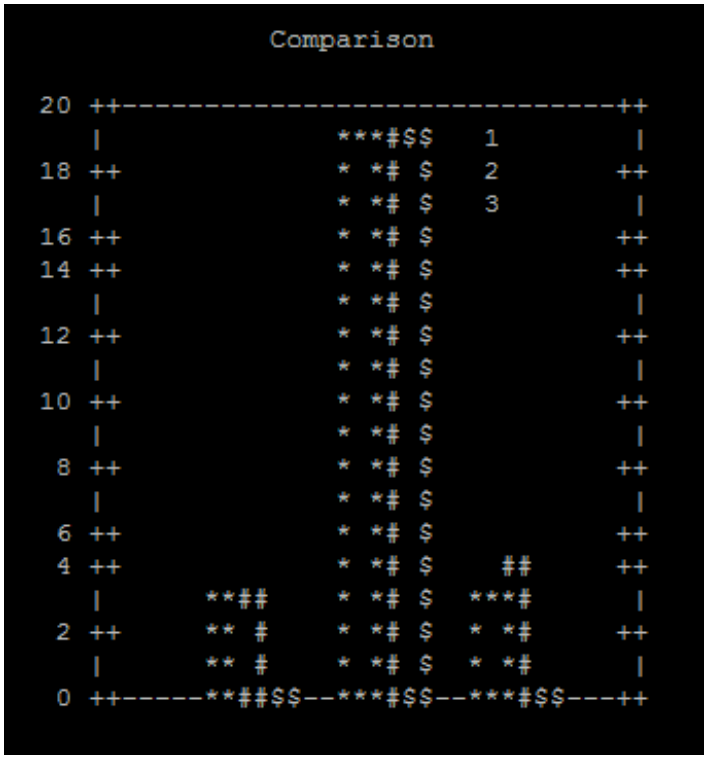
ext4		xfs	
8455.92	+1.8%	8609.96	kvm_lkp/micro/dd-write/
8462.31	+1.7%	8608.62	kvm_lkp/micro/dd-write/
8456.47	+1.8%	8611.48	kvm_lkp/micro/dd-write/
25374.70	+1.8%	25830.06	TOTAL iostat.vdb.wkB/s

Figure : 比较分析 iostat.vdb.wkB/s

数据比较可视化

根据前面的比较数据进行作图，方便比较：





简单算法

在最初测试阶段，我们首先对某个指定的测试指标进行比较测试：

1. 只寻找 regression（即下降）
2. 提供前后两个 commit 的某项测试数据
3. 预先对某一次的 commit 的相同 config 进行多次测试并记录测试数据
4. 人工分析历史数据确定偏离阈值

在我们得到的数据中，我们可以获取任意一次测试的离散时间点上的所有监视数据，这里我们只是用整个测试过程中的监视指标平均值进行比较。

简单算法存在的问题及优化思路

存在问题

1. 有时是 patchset 出现问题而不是单个 patch 出现问题，此时应该挑出整段的 patch 而不是几个独立的 patch
2. 阈值由人工指定，当比较的指标增多时，工作量比较大
3. 不同的 config 的阈值可能不同

优化思路

1. 连续 commit 造成的问题在 10-bisect 递归算法中解决
2. 在相同 config 的前提下，利用所有正常的历史测试数据，作标准差 σ，取阈值

$$\text{threshold} = k\sigma$$

使得我们能有较高的置信度（k=3 时，置信度为 99.7%）

下一步工作

中期之后马上将要展开的工作：

1. 根据数据进一步修改参数使得性能下降的判断算法能够更好地工作
2. 实现 10-bisect 搜索算法

Thank you!

Q&A?