история одной оптимизации: теория и практика

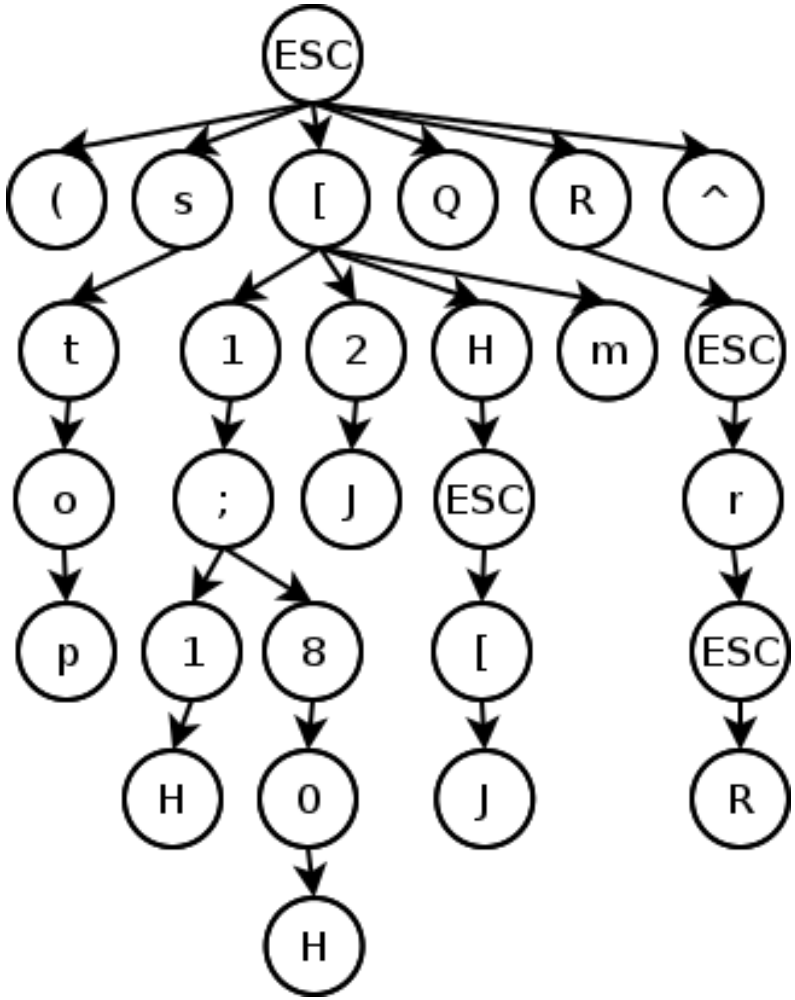
Maxim Gorbachyov <maxim.gorbachyov@gmail.com>

25 февраля 2012 г.

- кусочек IPMI stack: last boot log
- модуль ядра, кольцевой буфер, character device
- получаем данные по 1 байту
- иногда у источника данных сносит крышу и он генерит много ESC-последовательностей
- буфер ограничен, надо фильтровать поступающие данные

исходная реализация:

```
for (i = 0; i < patterns_count; ++i)
if (memcmp(pattern[i], current_buffer, current_size) == 0)
```



в теории эффективнее,
на практике тест показал время чуть хуже первого варианта

- "What Every Programmer Should Know About Memory" by Ulrich Drepper
- google: drepper cpumemory (cpumemory.pdf есть в наличии)
- а также другие работы автора

- в первом варианте:
  simple access pattern, данные лежат одним куском в памяти ->
  cache, hardware prefetch
- во втором варианте:
  for (i = 0; i < links_count; ++i)
  if (links[i]->ch == ch)
  доступ такого кода к нужным данным может быть медленнее (*)

- данные о следующих после текущего символах размещаем в одном куске памяти в узле.
- тест показал увеличение скорости работы: в первом варианте было 8 секунд, стало 3.

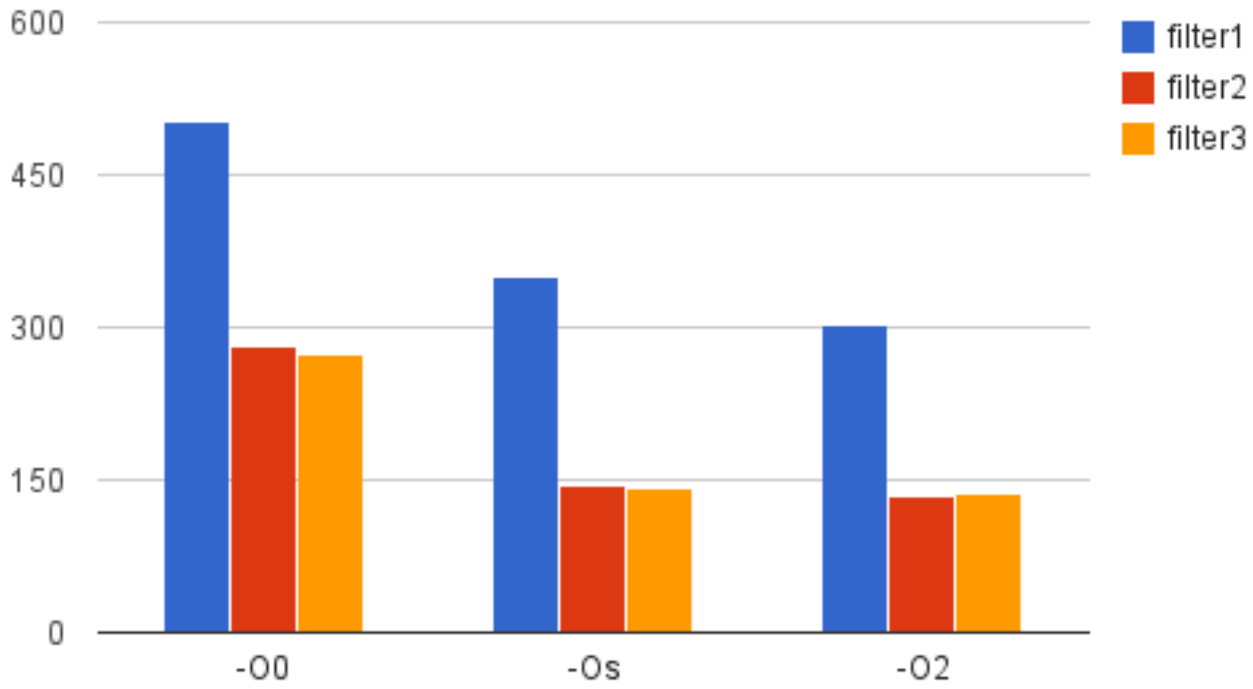
- воспроизвести результаты не удалось.
- разброс в результатах тестов для одного фильтра больше, чем для разных фильтров.
- вывод: что-то не так с тестами.

как НЕ НАДО делать:

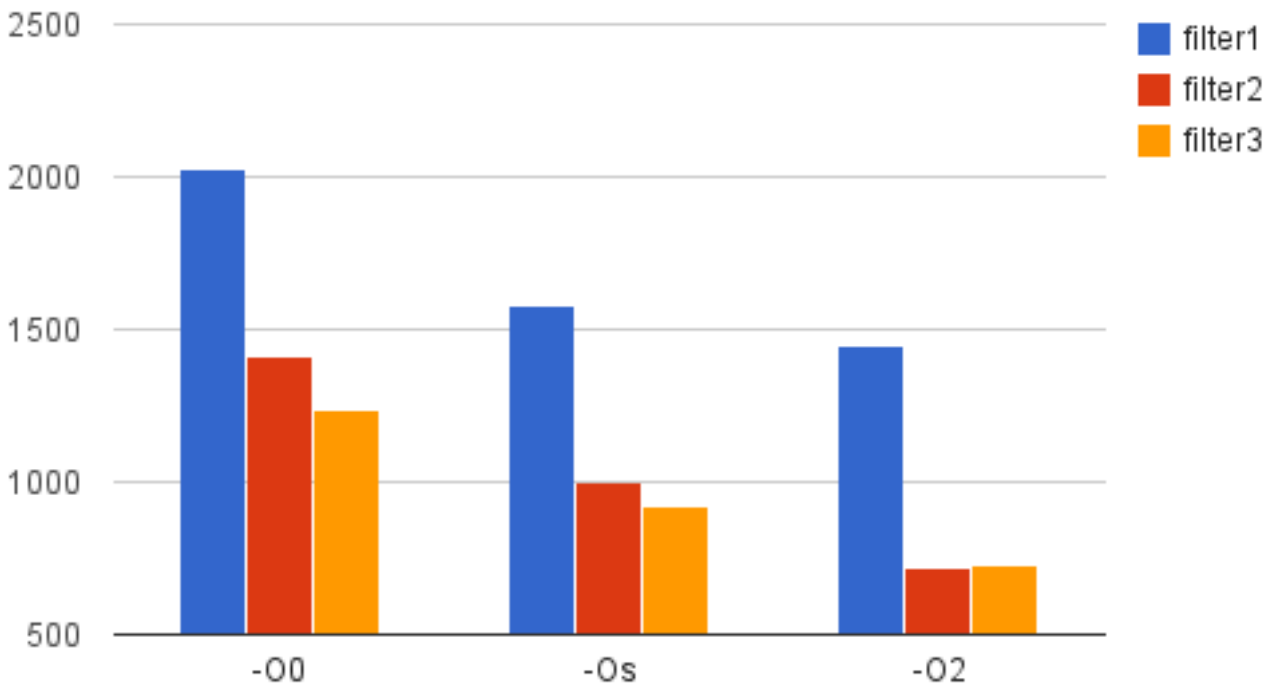
- while (read(fd, &ch, 1) == 1) detect(ch)
- time ./filtest 1 blob > /dev/null

итог: повторяемые результаты

Processor : ARM926EJ-S rev 5 (v5l)
BogoMIPS : 199.47
Features : swp half fastmult edsp java



model name : Intel(R) Core(TM)2 Duo CPU E6550 @ 2.33GHz
cache size : 4096 KB
bogomips : 4672.11
clflush size : 64
cache_alignment : 64



как -O2 удаётся убрать разницу между вариантами 2 и 3?