Optymalizacja programów Open-Source

Pamięć
część 2

Krzysztof Lichota
lichota@mimuw.edu.pl

Massif

Massif

- Narzędzie Valgrinda pozwalające sprawdzić w którym miejscu programu pamięć jest alokowana, kiedy i ile
- Pokazuje zużycie pamięci na stertę (heap), administację stertą i na stos
- Potrafi wygenerować wykres pokazujący zużycie pamięci w czasie

Zalety massifa

- Wykres pozwala szybko się zorientować, w którym miejscu alokujemy najwięcej pamięci
- Tekstowy raport pozwala zorientować się jaka jest ścieżka wywołania, która prowadzi do alokacji
- Bardzo łatwy w użyciu, nie wymaga rekompilacji programu

Wady massifa

- Obliczanie pamięci na administrowanie stertą jest przybliżone
- Nie można dokładnie oznaczyć miejsc w czasie, w których jakieś działanie się zaczyna lub kończy – trzeba w przybliżeniu zapamiętywać czas
- Program wykonuje się bardzo wolno
- Wyniki mogą być niedokładne, bo stan alokacji jest rejestrowany co jakiś czas

Użycie massifa

- Wywołanie: valgrind --tool=massif <polecenie>
- Po zakończeniu tworzony jest w bieżącym katalogu plik tekstowy zawierający opis ile pamięci z którego miejsca było zaalokowane
- Plik wynikowy należy przetworzyć narzędziem ms_print
- Domyślnie rejestrowane są dane co ileś wykonanych instrukcji, opcja --time-unit=B zmienia na rejestrowanie co ileś zaalokowanych bajtów (częstotliwość zrzutów jest dynamicznie zmieniana tak by zawsze trzymać <100)

Użycie massifa

- Jeśli używamy własnych funkcji alokujących pamięć można je wskazać massifowi za pomocą --alloc-fn
- Profilowanie stosu można włączyć opcją --stacks=yes (domyślnie wyłączona, bo bardzo spowalnia)
- Jeśli pokazywany stan stosu jest zbyt mały, by rozpoznać jednoznacznie miejsce alokacji, należy zwiększyć głębokość pokazywanego stosu za pomocą opcji --depth

Użycie ms_print

- ms_print generuje plik tekstowy z wykresem (ASCII) i szczegółowymi danymi zrzutów
- Pokazuje ilość zażądanej alokacji (useful-heap) i narzut administracyjny (extra-heap)
- Dla każdego snapshota pokazuje rozbicie z których funkcji (i ścieżek wywołania) nastąpiły alokacje i jaka wielkość
- Można zmienić parametry generowania wykresu
- „:” na wykresie oznacza zwykły snapshot, „@” szczegółowy, a „#” moment największego zużycia

kmtrace

kmtrace

- Narzędzie z pakietu KDE SDK
- Zawiera bibliotekę ładowaną za pomocą LD_PRELOAD i skrypty do przetwarzania wygenerowanych w ten sposób wyników
- Zbieranie i generowanie wyników nie wymaga GUI, więc można używać go do śledzenia demonów lub automatycznego zbierania wyników

kmtrace

- Domyślne skrypty przetwarzające służą do znajdowania wycieków pamięci, a nie do analizy zużycia pamięci
- Plik śladu zawiera rozmiar alokacji i zrzut stosu, więc można sobie go łatwo przetwarzać na swoje sposoby

Wady kmtrace

- Nie pokazuje zużycia stosu
- Nie pokazuje pamięci zaalokowanej za pomocą mmap()
- Nie uwzględnia fragmentacji pamięci
- Z tych powodów zużycie pamięci pokazywane przez memprof dość odbiega od prawdziwego zużycia po uwzględnieniu załadowanych bibliotek, stosu, itd.

Użycie kmtrace

- Uruchamiamy program za pomocą: MALLOC_TRACE=./ktrace.out LD_PRELOAD=/usr/lib/kmtrace/libktrace.so <polecenie> lub kminspector <polecenie>
- Ślad jest w pliku ktrace.out i ma format tekstowy
- Analiza śladu (wycieki): kmtrace <plik-śladu>
- Można zliczyć rozmiar i liczbę alokacji prostym skryptem:
  - cat ktrace.out | grep '^\+' | (s=0; i=0; while read dummy addr len; do i=$[$i + 1]; s=$[$s + len]; done; echo $i $s ;)

Bibliografia

- http://valgrind.org/docs/manual/ms-manual.html
- http://developer.gnome.org/doc/guides/optimisation/M
- http://ktown.kde.org/~seli/memory/analysis.html