Optymalizacja programów Open-Source

Profilery niskiego poziomu
część 2

Krzysztof Lichota
lichota@mimuw.edu.pl

oprofile

oprofile

- Działa na zasadzie odczytywania pozycji w kodzie i stanu stosu, gdy sprzętowe rejestry procesora dotyczące różnych zdarzeń osiągną określoną wartość (generowane jest przerwanie)
- Umożliwia badanie sprzętowych zdarzeń, które inaczej nie są możliwe do zbadania
- Umożliwia uzyskanie wyników z dość dużą dokładnością
- Umożliwia uzyskanie grafu wywołań

Zalety oprofile

- Pozwala na profiling jądra systemu, w tym procedur obsługi przerwań
- Mały narzut dzięki wykorzystaniu sprzętowych liczników (zwykle 1-3%)
- Pozwala na profiling bardzo niskopoziomowy (np. nietrafienia w cache CPU, złe branch prediction) na podstawie faktycznego działania procesora

Wady oprofile

- Wymaga uprawnień administratora
- Nie zapewnia 100% dokładności na poziomie instrukcji (ze względu na próbkowanie)
- Działa tylko na niektórych architekturach (m.in. i386/x86_64)
- Wyniki są trudne w interpretacji – trzeba znać dobrze architekturę i sposób działania procesora

Liczniki

- Różne rodzaje liczników, w zależności od procesora
- Najciekawsze (dla Pentium IV):
  - BRANCH_RETIRED – ominięte gałęzie wykonania
  - BSQ_CACHE_REFERENCE – trafienia/nietrafienia w cache różnych poziomów
  - MACHINE_CLEAR – opróżnienia pipeline procesora
  - RETIRED_MISPRED_BRANCH_TYPE – gałęzie wykonywania pominięte, podzielone według typu
  - PAGE_WALK_TYPE – dostępy do tablic stron
  - 64BIT_MMX_UOP – liczba mikrooperacji MMX
- Pełna lista: http://oprofile.sourceforge.net/docs/

Obsługa

- opcontrol --start --no-vmlinux – rozpoczęcie profilowania (bez jądra, z domyślnymi licznikami)
- opcontrol --shutdown – zatrzymanie profilowania
- opcontrol --dump – zrzut obecnego stanu, żeby można go było przetwarzać
- opcontrol --save nazwa – zapisanie sesji profilowania
- opreport – generuje raporty zbiorcze ze śladu
- opannotate – generuje anotację do kodu źródłowego lub asemblera

Obsługa (2)

- Zdarzenia do śledzenia podaje się przy uruchomieniu demona, za pomocą opcji --events
- Podaje się nazwę licznika i maskę zdarzeń (o ile jest potrzebna)
- Przykład: --event=DATA_MEM_REFS:30000
- Nie wszystkie kombinacje zdarzeń są możliwe
- Listę zdarzeń można uzyskać za pomocą wywołania: opcontrol --list-events
- Domyślnie na i386 jest CPU_CLK_UNHALTED
- Najlepiej użyć GUI: oprof_start

Przydatne opcje

- opreport --exclude-dependent --demangle=smart --symbols biblioteka – pokazuje wyniki z podziałem na symbole dla podanej biblioteki/pliku wykonywalnego
- opreport --demangle=smart --symbols plik-exe – pokazuje wyniki dla programu wykonywalnego i jego bibliotek
- opreport --long-filenames – pokazuje wyniki z podziałem na biblioteki/pliki wykonywalne
- opannotate --source --assembly plik-exe – tworzy anotację kodu źródłowego z wplecionym asemblerem

Użycie z kcachegrind

- Kcachegrind ma możliwość wyświetlenia wyników oprofile, dzięki czemu łatwiej można oglądać wyniki
- Należy skonwertować wyniki oprofile do postaci znanej Kcachegrind za pomocą polecenia: opreport -gdf | op2calltree
- op2calltree jest częścią pakietu kcachegrind (są również skrypty do konwersji z innych formatów, np. gprofa)

Na co zwrócić uwagę

- Ustawienie bardzo niskiego okresu zbierania danych może spowodować zamrożenie systemu
- Wartość graniczna licznika nie powinna być „równa”, żeby licznik nie zgrał się w fazie z naszym programem

Inne profilery

- Komercyjne
  - Rational Quantify
  - Intel VTune
- Wolnodostępne
  - HPCView
  - MemSpy (Martonosi/Gupta/Anderson)
  - CProf (Lebeck/Wood)
  - VProf
  - MTools (Goldberg/Hennessy)
  - MHSim (Fowler/Mellor-Crummey/Whalley)
  - SIP (Berg/Hagersten)
  - DCPI (HP Profiling Infrastructure for Alpha Processors)
- Źródło: http://kcachegrind.sourceforge.net/cgi-bin/show.cgi/KcacheGrindW

Instrukcje MMX/SSE

Instrukcje MMX/SSE

- Zestawy instrukcji do obrabiania dużych ilości danych, najczęściej multimedialnych (obraz, dźwięk), ale przydaje się też w innych sytuacjach, gdy obrabiamy dane strumieniowo
- Zawiera instrukcje SIMD (Single Instruction Multiple Data – zwane również wektorowymi) oraz inne przydatne instrukcje, np. do kontroli cache procesora, konwersji danych

Instrukcje SIMD

- Jedna instrukcja operuje na wielu danych jednocześnie, wykonując tę samą operację (np. xor, dodawanie, porównywanie, mnożenie, dzielenie, max)
- Większość instrukcji działa „pionowo” (np. dodajemy 4 słowa 2-bajtowe z jednego rejestru do 4 słów 2-bajtowych z 2 rejestru)
- Niektóre instrukcje działają „poziomo” (np. dodajemy 4 słowa 2-bajtowe w jednym rejestrze)
- Instrukcje mogą używać liczb całkowitych lub zmiennoprzecinkowych

Instrukcje kontroli cache

- Pozwalają z wyprzedzeniem wczytać do cache potrzebne dane i pozwalają określić do którego poziomu cache wczytać – PREFETCHTx
- Pozwalają określić, że dane będą użyte raz i nie należy trzymać ich w cache (zapobiegają cache pollution przy obróbce dużej ilości danych) – PREFETCHNTA
- Pozwalają wyrzucić z cache określone linie pamięci – CLFLUSH

Na co zwrócić uwagę

- Nie każdy procesor obsługuje dany zestaw instrukcji, należy zrobić przynajmniej wersję SIMD i zwykłą (w C)
- Instrukcje MMX są obsługiwane przez praktycznie każdy procesor dzisiaj w użyciu, więc najbezpieczniej pisać w tym zestawie instrukcji
- Niektóre kompilatory (np. GCC 4) są w stanie wygenerować kod w MMX, pod warunkiem użycia odpowiedniej docelowej architektury
- Czasem kod w C lub za pomocą wcześniejszego zestawu instrukcji jest szybszy!

Na co zwrócić uwagę (2)

- Po użyciu instrukcji MMX (nie SSE) należy wykonać instrukcję EMMS, ponieważ MMX używają rejestrów zmiennoprzecinkowych koprocesora
- Jeśli rozmiar danych nie jest wielokrotnością ilości danych przetwarzanych w naszym kodzie, należy dodać „ogon”, który przetworzy te ostatnie dane
- Dane powinny być wyrównane w pamięci do wielokrotności rozmiaru linii cache (16 bajtów), można do tego użyć funkcji posix_memalign() lub samemu alokować więcej i wyrównywać do wybranej granicy.

Bibliografia

- http://oprofile.sourceforge.net/
- http://en.wikipedia.org/wiki/MMX
- http://en.wikipedia.org/wiki/SSE2
- http://en.wikipedia.org/wiki/Automatic_vectorization
- http://nasm.sourceforge.net/doc/html/nasmdocb.html
  - lub jak nie działa: http://209.85.135.104/search?q=cache:_O4h0Dsudps
- http://www.ibiblio.org/gferg/ldp/GCC-Inline-Assembly-HO
- https://oa.doria.fi/bitstream/handle/10024/35105/nbnfi-fe2
- http://software.intel.com/en-us/articles/using-intel-vtune-p
- (3.4.1) http://www.intel.com/Assets/PDF/manual/248966.pdf