Optymalizacja programów Open-Source

Pamięć
część 1

Krzysztof Lichota
lichota@mimuw.edu.pl

Dlaczego pamięć jest ważna

- Współczesne systemy operacyjne stosują wirtualizację pamięci i symulują większą pamięć poprzez swapowanie
- Czas dostępu do dysku jest kilka rzędów wielkości wolniejszy od dostępu do pamięci
- Efektywny czas dostępu do pamięci:
  częstość_braku_strony*czas_sprowadzenia_strony + (1 – częstość_braku_strony) * czas_dostępu_do_pamięci

Dlaczego pamięć jest ważna

- Jeśli nasz proces zużywa dużo pamięci, to inne procesy mają mniej i cały system działa wolniej
- Każdy proces ma swoje pole robocze – zbiór stron, które są mu niezbędne do sensownego działania
- Przypadek skrajny – migotanie (thrashing), występuje kiedy suma pól roboczych programów w systemie jest większa niż dostępna pamięć
- Dużo wcześniej system zaczyna się dławić, ponieważ do sprawnego działania system potrzebuje też pamięci podręcznej dysku

Pamięć w systemie

Podział pamięci w systemie operacyjnym

- Pamięć dla programów
- Pamięć podręczna dysku (cache) – do przyspieszania odczytu
- Bufory na zapisane dane – przyspieszają zapis poprzez opóźnienie go do momentu aż system będzie miał wolne zasoby
- Pamięć jądra – struktury jądra, które muszą być cały czas w pamięci (np. kod do obsługi błędów braku strony) oraz pamięć wirtualna jądra, która może zostać wymieciona na dysk

Program free

- Pokazuje zużycie pamięci:
  - free – wolna pamięć (powinna być mała)
  - cached – pamięć podręczna dysku
  - buffers – bufory
  - -/+ buffers/cache – pamięć po odjęciu buforów i cache (czyli to, co tak naprawdę jest interesujące)
- Dokładniejsze dane: /proc/meminfo

vmstat

- Podaje interesujące statystyki zużycia pamięci, komunikacji ze swap i dyskiem oraz zużycia CPU
  - standardowe parametry pamięci (wolna, cache, itp.)
  - si/so – ilość pamięci ściągnięta/wysłana do swapa
  - bi/bo – ilość danych ściągnięta/wysłana na dysk
  - standardowe parametry CPU (user, kernel, iowait)
  - cs – liczba przełączeń kontekstu na sekundę
  - in – liczba przerwań na sekundę
- Wywołanie „vmstat 1” powtarza wyświetlanie co 1s
- Do rysowania wykresów można użyć OpenOffice lub automatycznie generować za pomocą gnuplot

Pamięć w programie

Pamięć w programie

- Dane użytkowe (sterta – ang. heap)
- Stos
- Kod bibliotek i programu
- Pliki wmapowane za pomocą mmap()
- Część pamięci używanej przez program (piksmapy) jest tak naprawdę w X-serwerze!

Pamięć w programie

- Czasem pamięć jest automatycznie współdzielona między procesami przez system operacyjny:
  - kod programu, wmapowane pliki tylko do odczytu, biblioteki dynamiczne
  - cała pamięć procesu przy fork() (poprzez copy-on-write), nawet stos!
- Kod bibliotek i programu jest zasadniczo współdzielony, ale relokacja symboli może spowodować, że część nie będzie współdzielona, tylko prywatna

Pamięć w programie

- RSS (Resident Set Size) – liczba stron w pamięci używanych przez program (łącznie ze współdzielonymi)
- VmSize – rozmiar pamięci wirtualnej procesu
- Shared – rozmiar pamięci potencjalnie współdzielonej z innymi procesami
- Te statystyki mogą być bardzo mylące, jeśli próbuje się ich użyć do podsumowania całego systemu lub dokładniejszego porównania programów [1]

[1] http://www.kdedevelopers.org/node/1567

time

- Podaje statystyki z uruchomienia procesu
- Wywołanie: /usr/bin/time <opcje> polecenie
- Opcja „-f format” pozwala wybrać statystyki do pokazania
- Przydatne statystyki do analizy pamięci:
  - %F – liczba „major” page faults
  - %R – liczba „minor” page faults
  - %M – maksymalny RSS w czasie życia programu
  - %p – średni rozmiar niewspółdzielonego stosu
  - %t – średni RSS procesu

ps

- Pokazuje podobne statystyki podobne jak time, ale dla wszystkich procesów i nie wymaga uruchamiania programu pod nadzorem
- Można wypisywać tylko jeden proces za pomocą opcji „-p pid”
- Jeśli zależy nam na interaktywnym wyświetlaniu statystyki, to można użyć programu „top”
- Więcej opcji: man ps

/proc

- Dużo informacji o procesach można bezpośrednio z /proc – ps i top używają go zresztą do pobierania informacji
- Ciekawe pliki w /proc/pid:
  - status – stan procesu, podsumowanie zużycia pamięci
  - statm – dokładniejsze informacje o pamięci procesu
  - maps/smaps – informacje o obszarach pamięci wirtualnej procesu
- Żeby odczytać dane o bieżącym procesie można użyć dowiązania symbolicznego /proc/self
- Specyficzne dla Linuksa (dlatego lepiej używać ps)

/proc/pid/maps

- /proc/pid/maps pokazuje obszary pamięci wirtualnej w procesie
- Flagi:
  - r/w – do odczytu/zapisu
  - x – do wykonywania
  - p – prywatne (mogą być współdzielone przez copy-on-write)
  - s – współdzielone explicite
- Biblioteki są wmapowane 2 razy – raz jako wykonywalne i raz jako do zapisu (żeby zrobić relokacje)
- Na i386 prawo wykonywania implikuje prawo odczytu

/proc/pid/smaps

- Pokazuje szczegółowe statystyki dla każdego obszaru pamięci wirtualnej
  - RSS – liczba stron w pamięci
  - Shared_Clean – współdzielone, które można w każdej chwili wyrzucić, bo są zapisane
  - Shared_Dirty – współdzielone, które zawierają niezapisane zmiany i które przed wyrzuceniem trzeba zapisać
  - Private_Clean – prywatne, które można wyrzucić
  - Private_Dirty – prywatne, które trzeba zapisać
- Tutaj „Shared” oznacza, że jest więcej niż jeden proces, który posiada strony w swoich tablicach stron!

exmap

- Pokazuje szczegółowe statystyki zużycia pamięci
- Potrafi pokazać statystyki pamięci dla procesu/biblioteki/sekcji ELF/symbolu
- Używa modułu jądra do uzyskania statystyk
- Ma interfejs graficzny
- Może pokazywać odwrotne mapowanie – z plików do procesów
- Pokazuje ilość współdzielonej pamięć rozdzieloną proporcjonalnie pomiędzy wszystkie używające jej procesy

xrestop

- Pokazuje zużycie pamięci w X-serwerze dla poszczególnych programów
- Używa rozszerzenie X-serwera X-resource
- Domyślnie pokazuje podobny widok jak „top”
- Opcja -b pozwala zrzucać dane wsadowo
- Uwaga – jak każdy program X pokazuje zużycie serwera X wskazywanego przez zmienną DISPLAY, więc jak podłączamy się zdalnie do komputera, to pokaże wyniki dla naszego lokalnego X-serwera.

Ukryte problemy z używaniem pamięci

- Wycieki pamięci – powodują zwiększanie objętości programu w trakcie działania
- Fragmentacja – powoduje, że program zaczyna zużywać więcej pamięci niż tak naprawdę potrzebuje
- Rozdzielczość stronicowania to rozmiar jednej strony (na i386 – 4 KB), więc każda operacja tworząca prywatną kopię strony powoduje zużycie co najmniej 1 strony, a jak struktura jest na granicy stron, to co najmniej dwóch

Co można optymalizować

- Znaleźć dane, które są w pamięci niepotrzebnie
- Znaleźć dane, które są duplikowane (np. jednocześnie w pamięci procesu i X-serwerze)
- Trzymać tylko część danych w pamięci potrzebną w danej chwili (programy graficzne, edytory tekstu)
- Użyć bardziej kompaktowych struktur danych
- Usunąć fragmentację
- Zmniejszyć zużycie czasoprzestrzenne – jeśli dane są konieczne, to trzymać je najkrócej jak się da
- Usunąć wycieki pamięci (to bardziej debugowanie)

Co można optymalizować

- Lepiej zarządzać danymi mmapowanymi (np. wyrzucać niepotrzebne za pomocą madvise)
- Współdzielić co się da (np. pliki konfiguracyjne, ikony), np. używając mmap() do danych niezmienianych lub pamięci dzielonej dla generowanych
- Ulepszyć współdzielenie kodu poprzez eliminację relokacji
- Zmniejszyć rozmiar danych aktywnie używanych w pamięci poprzez skompaktowanie ich na mniejszej liczbie stron
- Nie linkować się z niepotrzebnymi bibliotekami

Na co zwrócić uwagę

- Każdy wątek ma swój stos w trybie użytkownika (ok. 1MB domyślnie) – więcej wątków oznacza więcej zużytej pamięci (ale może być współdzielona przez copy-on-write)
- Każdy wątek/proces ma swój stos w trybie jądra (niestronicowany), o rozmiarze 4 KB lub 8 KB

Bibliografia

- http://bmaurer.blogspot.com/2006/03/memory-usage-with
- http://www.kdedevelopers.org/node/1567
- http://lxr.linux.no/source/Documentation/filesystems/proc
- http://monkey.linuxworld.com/cgi-bin/dwww?type=runma
- http://lists.kde.org/?l=kde-devel&m=116786313023874&
- http://www.berthels.co.uk/exmap/
- http://primates.ximian.com/~federico/docs/2007-02-FOSD
- http://people.redhat.com/berrange/olpc/performance/epip