Valgrind

Łukasz Matczyński

Wydział Fizyki, Astronomii i Informatyki Stosowanej

22.03.2012

Spis treści

Droga do Valhalii

Składniki Valgrinda

1. Memcheck
2. Cachegrind
3. Callgrind
4. Helgrind
5. DRD
6. Massif
7. SGCheck
8. BBV

Podstawowy sposób użycia

```
valgrind [valgrind-options] your-prog [your-prog-options]
```

przykład:

```
valgrind --tool=memcheck ls -l
```

Output

schemat informacji wyświetlanych przez valgrind'a

```
==12345== some-message-from-Valgrind
```

- 1234 - id procesu
- Dodatkowo opcja `-v` wyświetla bardziej szczegółowe informacje.
- Powtarzające się błędy nie są wyświetlane.
- Gdy liczba różych błędów wyniesie 100 - ostrzerze nie: `More than 100 errors detected. Subsequent errors will still be recorded, but in less detail than before.` - w celu określenia czy błędy są różne wykorzystuje od tego momentu stan licznika programu z dwóch ostatnich pozycji stosu.

Valgrind przestaje raportować błedy gdy:

1. Wykryje 1000 różnych błędów.
2. Całkowita liczba błędów wyniesie 10 000 000.

Wyświetlanie informacji o błędach

1. Domyślnie : stderr (deskryptor pliku 2), można zmienić deskryptor pliku - `--log-fd=9`.
2. Zapis do pliku - `--log-file=filename`.
3. Na zdalnej maszynie - `--log-socket=192.168.0.1:12345` , `192.168.0.1:12345` - IP:nr portu, na zdalnej maszynie musi działać `valgrind-listner`

Ustawienia domyślnych opcji

Opcje domyślne ustawiane w :

1. plik: `/.valgrind.rc`
2. zmienna sys: `$VALGRIND_OPTS`
3. plik: `./.valgrind.rc`

`$VALGRIND_OPTS` nadpisuje `/.valgrind.rc` itd.
Plik `./.valgrind.rc`, musi należeć do danego użytkownika, bez praw do zapisu przez innych.

Example

```
--memcheck:leak-check=yes
```

Ignorowanie błędów

Zastosowanie

Memcheck - memory errors detector

Wykrywa:

1. Dostęp do pamieci do której nie powiniśmy mieć dostępu : przekroczenie bloku sterty, przekroczenie górnej granicy stosu, czytanie pamięci po jej zwolnieniu.
2. Użycie nie zainicjalizowanych zmiennych.
3. Nie odpowiednie zwolnienie pamięci (np zwolnienie już zwolnionej pamięci)
4. Zamiana src i dst w funkcjach takich jak np. `memcpy`.
5. Wycieki pamięci.

Wycieki pamięci

Memcheck sprawdza czy blok pamieci jest osiągalny ze wskaźników root-setu (rejestry generalnego przeznaczenia wszystkich wątków + dostępna pamięć łacznie ze stosem).
Blok pamięci może być dostępny przez wskaźnik do jego początku lub wnętrza.

```
   Pointer chain         AAA Category   BBB Category
   -------------         ------------   ------------
(1)  RRR ------------> BBB               DR
(2)  RRR ---> AAA ---> BBB   DR          IR
(3)  RRR           BBB               DL
(4)  RRR     AAA ---> BBB   DL          IL
(5)  RRR ------?-----> BBB               (y)DR, (n)DL
(6)  RRR ---> AAA -?-> BBB   DR          (y)IR, (n)DL
(7)  RRR -?-> AAA ---> BBB   (y)DR, (n)DL   (y)IR, (n)IL
(8)  RRR -?-> AAA -?-> BBB   (y)DR, (n)DL   (y,y)IR, (n,y)IL, (_,n)DL
(9)  RRR     AAA -?-> BBB   DL          (y)IL, (n)DL

Pointer chain legend:
- RRR: a root set node or DR block
- AAA, BBB: heap blocks
- --->: a start-pointer
- -?->: an interior-pointer

Category legend:
- DR: Directly reachable
- IR: Indirectly reachable
- DL: Directly lost
- IL: Indirectly lost
- (y)XY: it's XY if the interior-pointer is a real pointer
- (n)XY: it's XY if the interior-pointer is not a real pointer
- (_)XY: it's XY in either case
```

- Still reachable - 1,2
- Definitely lost - 3
- Indirectly lost - 4,9
- Possibly lost - 5,8

Memcheck dodatkowe opcje

Pokazuje (yes i full) dodatkowe informacje o blokach pamieci

`--leak-check=<no|summary|yes|full> [default: summary]`

Pokazuje (yes) bloki reachable i indirectly lost

`--show-reachable=<yes|no> [default: no]`

Sledzi (yes) niezainicjalizowane zmienne.

`--track-origins=<yes|no> [default: no]`

Jak Działa Memcheck?

- Memcheck implementuje sztuczne CPU
- Każdemu bitowi odpowiada jeden bit V (valid-value)
- Niskopoziomowe operacje takie jak dodawanie są wykonywane dodatkowo na bitach V
- Sprawdzanie bitów tylko gdy:
  1. Wartość służy do obliczenia adresu pamięci
  2. Gdy wartość steruje przepływem - np instrukcja if
  3. Gdy wykonywane jest wywołanie systemowe
- Jeśli wartość jest nie określona zgłaszany jest błąd, a wynikowa wartość jest z powrotem określana jako zdefiniowana.

- Dodatkowo każdemu zwykłemu bitowi odpowiada bit A - valid-address bit.
- Określa on czy i kiedy program może czytać zapisywać daną lokację.
- Jest sprawdzany przy każdym zapisie/odczytcie.
- Gdy program się uruchamia wszystkie globalne zmienne są określone jako dostępne.
- Malloc - dostępne, free - nie dostępne.
- Obszar poniżej SP (wskaźnika stosu) - dostępny, powyżej nie dostępny.

Cachegrind

- Symuluje jak program działa z pamięcią cache i mechanizmem przewidywania rozgałęzień.
- Symuluje maszynę z pamięciami cache danych (I1 D1) i pamięcią cache drugiego poziomu L2 (LL last-level caches)
- Cachegrind zbiera statystyki:
  1. I cache reads (Ir, równa ilości wykonanych instrukcji), I1 cache read misses (I1mr) and LL cache instruction read misses (ILmr).
  2. D cache reads (Dr, równa ilości przeczytanej pamięci), D1 cache read misses (D1mr), and LL cache data read misses (DLmr).
  3. D cache writes (Dw, równa ilości zapisanej pamięci), D1 cache write misses (D1mw), and LL cache data write misses (DLmw).
  4. Conditional branches executed (Bc) and conditional branches mispredicted (Bcm).
  5. Indirect branches executed (Bi) and indirect branches

Cachegrind zapisuje informacje do pliku o nazwie `cachegrind.out.<pid>`. Nazwę pliku zmieniamy za pomocą opcji `--cachegrind-out-file` .
Dodawtkowe informacje na temat działania całego programu i poszczególnych zawartych w nim funkcji możemy otrzymać

Example

```
cg_annotate <nazwa pliku>
```

Między innymi informacje o symulowanym cachu - który określany jest na bazie instrukcji x86 CPUID
Dodatkowo dysponuje narzędziami takimi jak - `cg_merge`, `cg_diff`.

Callgrind

Pozwala zapamiętywać historię wywołań kolejnych funkcji i prezentować ją w postaći grafu (call-graph). Dodatkowo podobnie jak Cachegrind wykonuje symulacje cachey i rozgałeźień. Użycie
Do podgladania działającej symulacji callgrinda można użyć
`callgrind_control -e -b`
gdzie opcja e pokazuje ilość zdarzeń Instruction read.
Wyniki callgrind zapisuje do pliku, który możemy podejżeć za pomocą `callgrind_annotate` lub korzystając z `kchacegrinda`.

Hellgrind

Służy do wykrywania błędów synchronizacji w programach używających modelu wątków zgodnych z normą POSIX. Pozwala wykrywać zakleszczenia, wyścigi, nieodpowiednie użycia mutexów. Hellgrind podaje informacje o:

- Punktach w których powstały wątki.
- O adresie pod którym prawdopodobnie występuje adres.

Massif

Pozwala określić jaką ilość pamięci sterty wykorzystuje program. Może także mierzyć rozmiar stosu programu. Podawać informacje jakie części programu jaką część pamięci używają.

- Tak jak np Cachegrind Massif wyniki swoich działan zapisuje do plik `massif.out.pid`.

- Plik możemy oglądać za pomocą `ms_print`.
- Massif wykonuje graf wykorzystania pamięci - domyślna jednostka czasu - liczba instrukcji. Opcja `--time-unit=B` pozwala zmienić ją na liczbę zaalokowanych/zwolnionych bajtów.

Wykres

Massif wykonuje początkowo zaznaczenie punktu na wykresie przy każdej allokacji/deallokacji danych (przy 100 razie zmniejsza częstość).

- ":" - normalne dane - zapisywane są podstawowe informacje
- "" - dokładne informacje - czestosc zapisu ustala opcja `--detailed-freq`
- "#" - pik - oznacza miejsce największego zużycia pamięci, pobierany zawsze po deallokacji