Arm DDT

Wprowadzenie do Arm DDT

DDT to debugger, umożliwiający wykrywanie błędów, przerywanie działania programu, wykonywanie kodu krok po kroku, podglądanie wartości zmiennych, stosu wywołań funkcji itp., **dostosowany do pracy z aplikacjami rozproszonymi i wielowątkowymi**.

Niniejsze wprowadzenie zakłada znajomość: Tryb graficzny (przypomnienie).

Kiedy używać DDT (wskazówki)?

- jeśli w trakcie wykonania, program przerywa swoje działanie i kończy się niespodziewanym błędem
- jeśli program się zawiesza
- jeśli otrzymujemy niepoprawne wyniki i chcemy poznać przyczynę
- jeśli chcemy prześledzić działanie kodu

Korzystanie z debuggera ma największy sens gdy posiadamy dostęp do kodu źródłowego.

Jak uruchomić DDT?

Rozważmy przykładową aplikację, uruchamianą pod MPI:

```
mpiexec -n 4  ./mmult1_c.exe 1024
```

W celu jej zdebugowania wystarczy (pracując w trybie graficznym):

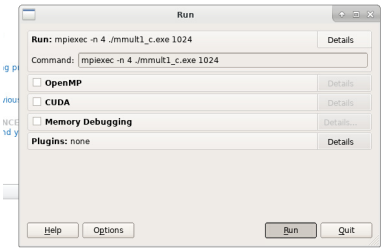
- załadować pakiet arm-forge

```
module add plgrid/tools/arm-forge
```

- skompilować program z flagą **-g** (symbole do debugu);
  bez tej opcji debuger nie będzie wiedział która linia jest wykonywana, być może będą znane nazwy funkcji
- zaleca się kompilację bez optymalizacji, tj. z flagą **-O0**;
  przy włączonych optymalizacjach (-O2/-O3), informacje z debugera mogą być zaburzone
- uruchomić program w ten sam sposób, **dodając na początku komendy "ddt"**

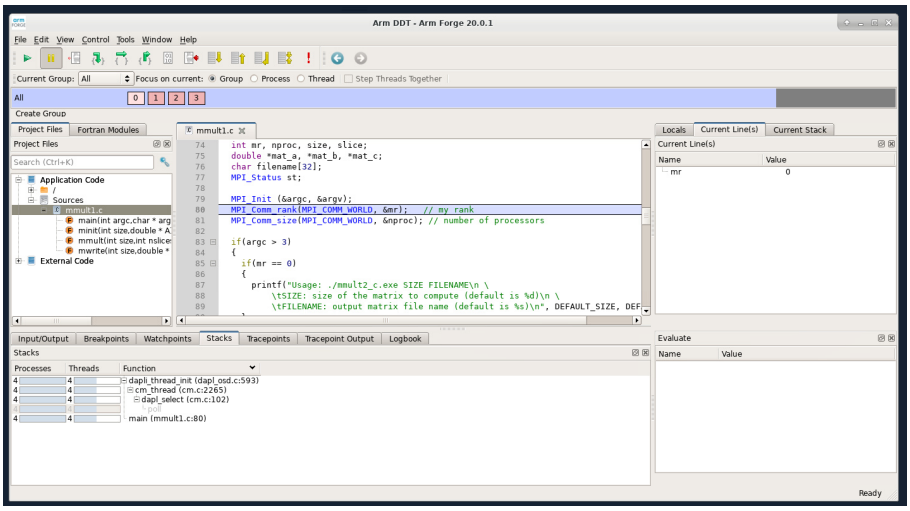
```
ddt  mpiexec -n 4  ./mmult1_c.exe 1024
```

W okienku, które się pojawi, klikamy RUN.



Otworzy się wtedy właściwe GUI programu DDT, w którym będziemy wykonywać proces debugowania.

Podstawowe funkcjonalności



- stack trace - wielowątkowy (gdzie się znajduje który proces/wątek)
- możliwość tworzenia grup procesów (pasek na górze)
- breakpointy
- watchpointy
- ewaluacja wyrażeń
- widok "LOGu" (zapis pracy debuggera)
- input i output programu
- podgląd wartości w danej linijce / podgląd zmiennych lokalnych (zestawienie z wszystkich procesów)
- kontrola wykonania programu (start, stop, step, step into, restart, ...)

Dodatkowe uwagi

- User Guide (wersja 20.2),
- **DDT z opcją offline,** umożliwia wykonanie kodu pod kontrolą debuggera, **bez interakcji użytkownika**; może to mieć zastosowanie gdy chcemy wykonać kod i podglądnąć wartości w konkretnym miejscu, albo gdy aplikacja kończy się błędem tylko raz na jakiś czas (wtedy spośród wielu wykonań, to które zakończy się błędem dostarczy nam informacji o błędzie); wartości zapisane w raporcie, odpowiadają temu co pojawia się w "Logbook"

```
ddt --offline=debug-report.html  --break-at=<file>:<line>   (...command...)
```

- **opcja attach**, umożliwia **podpięcie się do już uruchomionego procesu** (podając jego PID); bardzo wygodnie można to zrobić z poziomu GUI programu
- w DDT można otwierać pliki core-dump