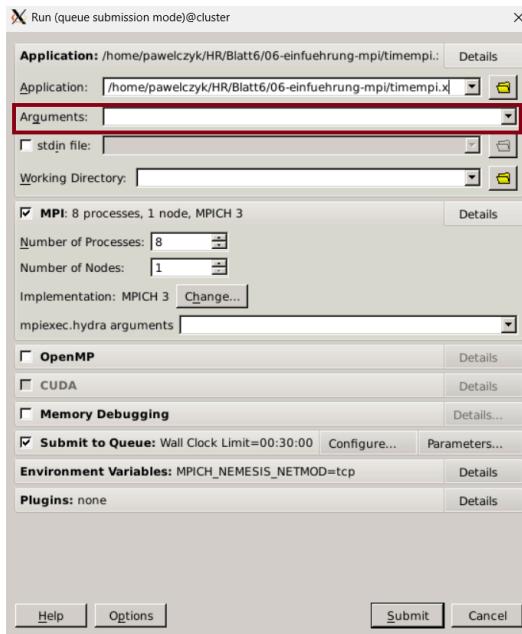


Wie kann man in DDT die Programmparameter angeben? Geben Sie zwei Wege an.

1. Über die Kommandozeile mit:

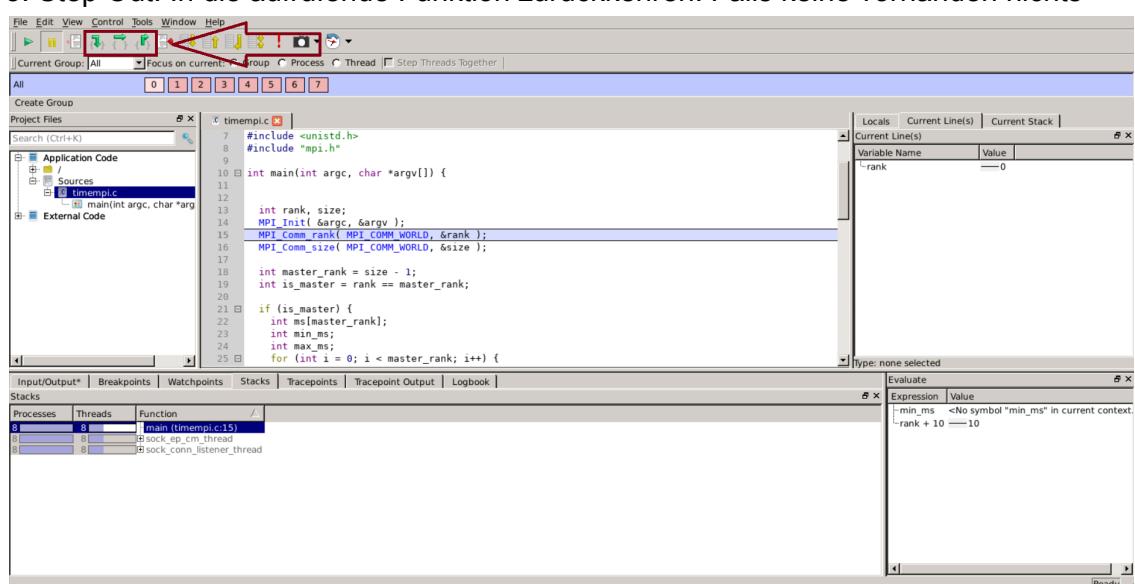
ddt ./programmname -- parameter1 parameter2 etc.

2. Nach der "Run" Option in das "Arguments" Feld eingeben



Setzen Sie in einer Zeile einen Breakpoint. Welche Step-Möglichkeiten gibt es und wie unterscheiden sich diese?

1. Step Into: Falls Funktionsaufruf vorhanden in die Funktion springen, sonst nicht
2. Step Over: Bedingungslos in die nächste Codezeile gehen
3. Step Out: In die aufrufende Funktion zurückkehren. Falls keine vorhanden nichts



Schauen Sie sich die Werte der Variable an, in der Sie den Rang des aktuellen Prozesses gespeichert haben. Erklären Sie die Linien in der Darstellung und vergleichen Sie die Werte aller Prozesse mit Hilfe des Kontextmenüs.

Die Linien symbolisieren jeweils Prozesse mit dem Rang des aktuellen Prozesses daneben

The screenshot shows a debugger interface with several windows:

- Project Files:** Shows a file named `timempic.c`.
- Locals:** A table showing variable values for the current line. It has two entries: `rank` with value `0` and `size` with value `4199629`. A red arrow points from the text "Die Linien symbolisieren jeweils Prozesse mit dem Rang des aktuellen Prozesses daneben" to the `rank` entry.
- Evaluate:** An expression evaluation window showing `rank + 10` with value `10`.
- Stacks:** A table showing processes, threads, and functions. It lists three processes (rank 8) with threads (rank 8) running in `main`, `sock_ep_cm_thread`, and `sock_conn_listener_thread`.

Machen Sie sich mit der Funktion des Evaluate-Fensters in der rechten unteren Ecke vertraut.

Im Expression-Fenster kann man Variablen oder Ausdrücke angeben und deren Zustände bzw. Ergebnisse mit aktuellen Werten während des Programms beobachten. Dabei hängen die Werte auch davon ab, in welchem Prozess man sich befindet.

The screenshot shows a debugger interface with several windows:

- Project Files:** Shows a file named `timempic.c`.
- Locals:** A table showing variable values for the current line. It has two entries: `rank` with value `0` and `size` with value `4199629`.
- Evaluate:** An expression evaluation window showing `rank + 10` with value `10`. A red arrow points from the text "Die Linien symbolisieren jeweils Prozesse mit dem Rang des aktuellen Prozesses daneben" to the `rank` entry.
- Stacks:** A table showing processes, threads, and functions. It lists three processes (rank 8) with threads (rank 8) running in `main`, `sock_ep_cm_thread`, and `sock_conn_listener_thread`.

In der oberen Leiste finden Sie eine Übersicht aller Prozesse und Threads Ihres Programmes. Wechseln Sie zwischen den einzelnen Prozessen und beobachten Sie das Evaluate-Fenster.

Variablen wie z.B. rank ändern sich beim Wechsel des Prozesses im Expression-Fenster. D.h. Werte und klar vom ausgewählten Prozess abhängig

The screenshot shows the DDT (DynaTrace Debugger) interface. On the left is a code editor with the file 'timempi.c' open, displaying MPI initialization and communication code. In the center is a stack viewer showing multiple threads (Processes 8, 9, 10) and their respective thread IDs. On the right is an 'Evaluate' window with a 'Locals' tab showing the variable 'size' with a value of 4199629. A red arrow points from the stack viewer towards the evaluate window, indicating that the variable's value is context-dependent on the selected thread.

Erweitern Sie Ihr Programm um ein Array und initialisieren Sie es mit beliebigen Zahlenwerten. Lassen Sie sich die Werte anzeigen. Visualisieren Sie dieses Array in DDT. (Hinweis: Das Array wird nur für diese Aufgabe benötigt und soll im abgegebenen Programm timempi nicht enthalten sein.)

This screenshot shows the DDT interface again, but with a modified version of 'timempi.c'. Line 17 now contains the assignment 'int test[4] = {rank + 1, rank + 2, rank + 3, rank + 4};'. The evaluate window on the right shows the variable 'test' with four entries: -test[0] <value has been optimized out>, -test[1] <value has been optimized out>, -test[2] <value has been optimized out>, and -test[3] <value has been optimized out>. A red arrow points from the stack viewer towards the evaluate window, similar to the previous screenshot, illustrating that the array's values are also context-dependent on the selected thread.