
	<p style="text-align: center;">Parallelrechner Übung Prof. Dr.-Ing. W. Rehm</p> <hr/> <p style="text-align: center;">Praktische Übung 4: Paralleles Debugging Bearbeiter: Nico Mittenzwey Letzte Änderung: René Oertel, 28.05.2014</p>	
---	--	---

1 Ziel

- Debuggen von parallelen Programmen

2 Einführung

Mit Hilfe eines Debuggers lassen sich Programme unterbrechen, Variablen zu unterschiedlichen Zeitpunkten anzeigen und ggf. auch modifizieren. Unter Linux ist der gdb (GNU Debugger) [2] der verbreitetste Debugger für serielle Programme. Da er für die Konsole entwickelt wurde, existiert auch eine u. a. von Red Hat entwickelte grafische Benutzerschnittstelle Insight [3].

Parallele Programme lassen sich mit gdb jedoch nur unter erheblichem Aufwand debuggen. Deshalb wurden von unterschiedlichen Firmen Debugger entwickelt, die auch parallele Programme unterstützen:

- *TotalView* von Rogue Wave Software
- *DDT* von Allinea

Beide sind jedoch proprietär und kostenpflichtig. Seit 2006 wird die auf Eclipse basierende *Parallel Tools Platform* (PTP) entwickelt. Ziel dieses Projektes ist es, eine Open Source Entwicklungsumgebung zur Verfügung zu stellen, die speziell auf das Entwickeln von parallelen Programmen ausgerichtet ist. Dabei unterstützt die Plattform ein breites Spektrum an parallelen Programmiermodellen (MPI, UPC, OpenMP), parallelen Architekturen und Laufzeitumgebungen. Zusätzlich bietet sie eine zugängliche Oberfläche zur Kontrolle von parallelen Systemen wie Clustern und enthält den parallelen Debugger sdm. Dieser basiert auf dem gdb, erlaubt aber durch Erweiterungen das einfache Debuggen von parallelen Prozessen innerhalb der Eclipse GUI.

3 Eclipse PTP - Parallel Tools Platform

Eclipse und PTP sind im RA-Pool zusätzlich, angepasst installiert und können ohne Installation genutzt werden. Der zugehörige Pfad lautet: /mnt/exports/software/eclipse/eclipse. Die folgenden Schritte sind deshalb nicht notwendig und dienen nur zur Referenz für private Tests. An dieser Stelle den vollständigen Funktionsumfang von PTP zu beschreiben, würde den Rahmen der Übung sprengen. Aus diesem Grund wird auf die Folien des „Full Day Tutorials“ zu PTP der Super Computer '09 Konferenz verwiesen: <http://wiki.eclipse.org/PTP/tutorials/SC09>.

3.1 Installation

1. Download von „Eclipse IDE for C/C++ Developers (87 MB)“ von www.eclipse.org/downloads, danach entpacken und starten
2. Über den Menüeintrag *Help* → *Install New Software...* in das „Install“-Fenster wechseln
3. Neues Repository über *Add...* hinzufügen:
 Name: CDT
 Location: <http://download.eclipse.org/tools/cdt/releases/helios/>

4. Neues Repository über *Add...* hinzufügen:
 Name: PTP
 Location: <http://ftp-stud.fht-esslingen.de/pub/Mirrors/eclipse/tools/ptp/releases/helios/>
 (Mirror von eclipse.org)
5. In *Work with PTP* auswählen, und
 - (a) gesamtes *Parallel Tools Platform* Paket selektieren
 - (b) *RSE Enabler*, *IBM LoadLeveler*, *IBM Parallel Environment* und *PBS* wieder deselektieren
6. Die ausgewählten Pakete installieren
7. Eclipse beenden

Kompilieren des Debuggers Der Debugger *sdm* liegt als C-Code vor und muss von Hand kompiliert werden. Dazu innerhalb des Eclipse-Verzeichnisses folgendes ausführen:

```
cd plugins/org.eclipse.ptp.linux.x86_64_3.0.1.201008031934
sh BUILD
```

Dadurch wird, neben anderen Tools und Proxys, der *sdm* kompiliert. Dieser ist danach im *bin*-Verzeichnis zu finden.

3.2 Resource Manager

Parallele Programme werden meist mit Hilfe von Hilfsprogrammen gestartet. Die Steuerung dieser übernimmt im PTP der Resource Manager. Im einfachsten Fall ist dieses Hilfsprogramm wie bei Open MPI ein auf dem Entwicklungssystem vorhandenes Programm (*mpiexec*). Ein Resource Manager kann aber auch ein Batchsystem (wie *PBS*, *SLURM* oder *IBM LoadLeveler*) nutzen, welches verteilte Programme auf einem Cluster startet.

Erstellen eines Open MPI Resource Managers

1. Öffnen der *Parallel Runtime* Perspektive über *Window* → *Open Perspective* → *Other...*, dort *Parallel Runtime* aus der Liste auswählen
2. Rechtsklick in das Fenster *Resource Managers* und dort *Add Resource Manager...* auswählen
3. *Open MPI* auswählen und auf *Next* klicken
 - (a) Im Dialog *Open MPI connection configuration* muss *Local* bei *Remote service provider*, sowie bei *Remote location* ausgewählt sein. *Tunneling Options* ist *None* und *Local address* ist *localhost*.
 - (b) Im Dialog *Open MPI tool configuration* kann Open MPI Version auf *Auto Detect* und *Use default location* ausgewählt gelassen werden.
 - (c) Im Dialog *Common Resource Manager Configuration*, sowie *Select Service Configuration* können die Standardeinstellungen verwendet werden.
 - (d) Nach einem Klick auf *Finish* sollte der neue Resource Manager in der Liste aller Resource Manager erscheinen.
4. Jetzt kann der Resource Manager durch einen Rechtsklick auf diesen und Auswahl von *Start Resource Manager* gestartet werden.

3.3 Erstellen eines neuen MPI-Projektes

Als Beispiel soll nun ein einfaches „Hello World“-MPI-Programm erstellt werden. Dazu ist Folgendes durchzuführen:

1. Öffnen der *C/C++*-Perspektive über *Window* → *Open Perspective* → *Other...* und dort *C/C++* auswählen
2. *File* → *New* → *C Project* und dort einen Namen für das neue Projekt eingeben

- (a) In der Liste *Project types*, *Executable* expandieren und *MPI Hello World C Project* auswählen
 - (b) Sollte hier ein Benachrichtigungsfenster erscheinen, mit der Frage, ob die MPI-Präferenzen gesetzt werden sollen, dieses bestätigen und die Voreinstellungen einfach übernehmen
 - (c) Im Dialog *Basic Settings* können die Standardeinstellungen verwendet werden
 - (d) Im Dialog *MPI Project Settings* muss, sollte etwas angezeigt werden, *Add MPI Project settings to this project* ausgewählt sein
 - (e) Beenden mit *Finish*
3. Jetzt sollte das Projekt im Workspace im Fenster *Project Explorer* sichtbar sein und automatisch kompiliert werden. Falls nicht, kann dies über *Project* → *Build All* getan werden. Zum Testen kann nun das Projekt über *Run* → *Run* gestartet werden.

3.4 Starten von parallelen Programmen

Zum Starten von parallelen Programmen wird ein Resource Manager benötigt. Das MPI-Projekt muss dazu entsprechend konfiguriert werden:

1. Öffnen der C/C++-Perspektive über *Window* → *Open Perspective* → *Other...* und dort C/C++ auswählen
2. Unter *Run* → *Run Configurations...*, *Parallel Application* auswählen, nach Rechtsklick auf dieses auf *New* klicken und im Feld *Name* einen Namen eingeben
3. *Resources* Tab
 - (a) Auswahl des Open MPI Resource Managers in der Drop-Down-Liste (dieser muss vorher gestartet wurden sein)
 - (b) Queue bleibt leer
 - (c) Zahl > 0 in das Feld *Number of processes* eintragen
4. *Application* Tab
 - (a) Das *MPI Hello World C Project* im Feld *Parallel Project* auswählen
 - (b) Den Pfad zum Arbeitsverzeichnis unter *Application program* auswählen (dieser sollte im RA-Pool unter */home/pool/shared/<nkz>* liegen)
 - (c) *Copy executable from local filesystem* auswählen
 - (d) Im *Path to the local file* die kompilierte Anwendung angeben (im Verzeichnis *debug* zu finden)
 - (e) *Display combined output in a console view* auswählen
5. *Debugger* Tab
 - (a) Debugger: *sdm*
 - (b) *Stop in main() on startup* auswählen
 - (c) *Path to debugger executable* ist im RA-Pool: */home/pool/shared/sdm* (andernfalls wie im Abschnitt 3.1 beschrieben)
 - (d) *Debugger session address: localhost*
6. *Arguments* Tab
 - (a) Hier können Programmargumente angegeben werden – für das „Hello World“-Beispiel werden jedoch keine benötigt
7. Mit *Apply* beenden
8. Nun kann das Programm mit einem Klick auf *Run* gestartet und in der *Parallel Runtime* Perspektive überwacht werden¹

¹Eine Legende aller Symbole erhält man durch einen Klick auf das 10. Symbol in der Toolbar.

3.5 Einrichten des Debuggers

1. Öffnen der C/C++-Perspektive über *Window* → *Open Perspective* → *Other...* und dort C/C++ auswählen
2. Öffnen der *Debug Configurations* über *Run* → *Debug Configurations...*, dort *Parallel Application* auswählen, nach Rechtsklick auf dieses auf *New* klicken und im Feld *Name* einen Namen eingeben
3. Danach wie im Abschnitt **Starten von parallelen Programmen** beschrieben weiter

4 Paralleles Debugging mit der Eclipse Parallel Tools Platform

Nach dem Laden eines Projektes und Einrichten des Debuggers, kann dieses in der *Parallel Debug* Perspektive debuggt werden. Der Debugvorgang kann dabei über *Run* → *Debug* oder die *Debug Configuration* gestartet werden.

4.1 Prozessgruppen: Erzeugung und Steuerung

Zum gleichzeitigen Steuern von mehreren Prozessen lassen sich Prozessgruppen definieren. Diese werden im Fenster *Parallel Debug* symbolisch als gelbe Vierecke angezeigt. Anfangs sind alle Prozesse in einer Gruppe. Über diesen befindet sich die Kontroll-Toolbar. Die ersten 6 Symbole dienen der Steuerung der angezeigten Prozesse. Mit den letzten 8 Symbolen lassen sich die Prozesse in Gruppen unterteilen.

Gruppieren von Prozessen Ein Prozess kann durch Anklicken ausgewählt werden. Mehrere Prozesse lassen sich mit Hilfe von Shift oder Ctrl auswählen. Außerdem existiert die Möglichkeit durch Gedrückthalten der linken Maustaste einen Rahmen um die gewünschten Prozesse zu ziehen. Danach kann durch einen Klick auf das $\{\}^+$ Symbol eine neue Prozessmenge² erstellt werden.

Um weitere Prozesse zu einer Menge hinzuzufügen, muss auf das v neben dem $\{\}^+$ Symbol geklickt und dort die Zielmenge ausgewählt werden. Über das letzte Symbol in der Toolbar lässt sich die jeweils zu steuernde Prozessmenge auswählen. Um Prozesse aus einer Menge zu entfernen, müssen diese ausgewählt und auf das $\{-\}$ Symbol geklickt werden. Das $\{\}^x$ Symbol löscht die gesamte Prozessmenge.

Steuern von Prozessmengen Über die ersten 6 Symbole lassen sich ganze Prozessmengen Fortfahren, Pausieren, Beenden und den Instruction Pointer im Code fortsetzen.

4.2 Breakpoints

Breakpoints können im Quellcodefenster durch einen Doppelklick auf die linke graue Leiste neben dem Code gesetzt werden. Dabei existieren zwei Arten von Breakpoints: Globale Breakpoints und Breakpoints für Prozessmengen.

Globale Breakpoints Globale Breakpoints können nur gesetzt werden, wenn im Fenster *Parallel Debug* kein Job ausgewählt wurde. Diese dienen vor allem dem Setzen von Einstiegspunkten ins Programm, sind in allen Prozessen gültig und durch einen grünen Kreis mit einem kleinen G in der oberen rechten Ecke gekennzeichnet. Sie bleiben bestehen, wenn ein Job beendet wird.

Breakpoints für Prozessmengen Breakpoints für Prozessmengen können für die gerade ausgewählte Prozessmenge (alle angezeigten gelben Vierecke im Fenster *Parallel Debug*) gesetzt werden. Diese Breakpoints haben – je nach Gruppenzugehörigkeit – eine bestimmte Farbe:

Grün Der Breakpoint bezieht sich auf die aktuelle Prozessmenge.

Gelb Der Breakpoint bezieht sich auf eine andere Prozessmenge.

²Hier soll das Wort „Menge“ eine Verwechslung mit den „Gruppen“ in MPI selbst vermeiden

Blau Der Breakpoint bezieht sich auf eine andere Prozessmenge, allerdings sind Prozesse aus dieser Menge auch in der aktuellen.

Wenn man den Mauszeiger über das Symbol für einen Breakpoint hält, erscheint nach kurzer Zeit ein Tooltip mit Detailinformationen. Im Fenster oben rechts lässt sich eine Übersicht über alle Breakpoints anzeigen.

4.3 Instruction Pointer

Der Instruction Pointer zeigt die aktuelle Position des angehaltenen Prozesses im Code. Dabei zeigt ein hellblauer Pfeil, die aktuelle Position eines registrierten Prozesses und ein dunkelblauer die eines unregistrierten Prozesses an. Wenn man den Mauszeiger über das Symbol für einen Instruction Pointer hält, erscheint nach kurzer Zeit ein Tooltip mit Detailinformationen.

4.4 Prozesse: Registrierung und Steuerung

Nachdem alle Prozesse eines Jobs angehalten wurden, lassen sich Detailinformationen über einen Prozess abrufen. Dazu muss dieser Prozess *registriert* werden. Es können auch mehrere Prozesse gleichzeitig registriert werden. Dazu muss auf den zu registrierenden Prozess im Fenster *Parallel Debug* doppelt geklickt werden. Wenn ein Prozess registriert ist, wird schwarzes Viereck um diesen angezeigt. Außerdem erscheint der Prozess im Fenster *Debug*. Hier kann nun durch einen Klick auf den *Stack Frame* (blau gestreiftes Viereck) der Prozess zur expliziten Steuerung ausgewählt werden. Außerdem wird nach der Auswahl im rechten oberen Fenster unter Variables der Inhalt aller Variablen angezeigt.

Durch einen weiteren Doppelklick auf ein registriertes Prozessviereck, lässt sich dessen Prozess wieder deregistrieren.

5 Aufgaben

5.1 Praktische Aufgaben

1. Richten Sie Ihren Workspace wie beschrieben (Resource Manager, Starten- und Debugkonfiguration) ein!
2. Machen Sie sich anhand des Hello World Beispiels mit dem Debugger vertraut!
3. Finden Sie mit Hilfe des Debuggers heraus, warum der Vorgabecode bei weniger als 5 Prozessen in einem Deadlock stehen bleibt!

Hinweise, Berichtigungen und Kritik zu den Übungsunterlagen bitte an:

- René Oertel <rene.oertel@cs.tu-chemnitz.de>

Literatur und wichtige Links

- [1] *MPI 2.2 Standard*
<http://www.mpi-forum.org/docs/docs.html>
- [2] *GDB (GNU Debugger)*
<http://www.gnu.org/software/gdb/>
- [3] *Insight (GUI für GDB)*
<http://sourceware.org/insight/>
- [4] *Eclipse*
<http://www.eclipse.org/>
- [5] *PTP - Parallel Tools Platform*
<http://www.eclipse.org/ptp/>