TSUNAMI

Testumgebung zur **s**ystematischen **U**ntersuchung **n**umerischer
Anwendungen und **m**athematischer **I**mplementationen

Benutzerhandbuch

(Version 1.0)

Martin Kreutz & Nils Nitsch

Inhaltsverzeichnis

1 Einleitung

Das Programm TSUNAMI (**T**estumgebung zur **s**ystematischen **U**ntersuchung **n**umerischer **A**nwendungen und **m**athematischer **I**mplementationen) wurde im Rahmen eines Systementwicklungsprojektes am Lehrstuhl für „Ingenieuranwendungen in der Informatik, numerische Programmierung“ von Herrn Professor Zenger an der Technischen Universität München entwickelt.

Es wird zum Ausführen und Auswerten von dort entwickelten numerischen Simulationen verwendet. Dabei übernimmt die TSUNAMI-Applikation das Kopieren auf etwaige Zielrechner, das Kompilieren des Programmcodes, geschrieben in der Programmiersprache C oder C++, die dortige Ausführung und das Zurückholen der Ausgabe- und Log-Dateien.

Der komplette Vorgang von der Kompilierung bis zur Auswertung wird im folgenden als ‚Testfall’ bezeichnet.

Die TSUNAMI Applikation ermöglicht es dem Benutzer mehrere Testfälle anzulegen und sie dann alle nacheinander, voll automatisch ausführen zu lassen. Dabei soll dem Benutzer die Möglichkeit gegeben werden, die Testfälle mit unterschiedlichen Konfiguration auszuführen.

Die TSUNAMI Applikation wurde unter Benutzung der Programmier-sprache JAVA™ (JDK 1.4) entwickelt. Zur Interaktion mit dem Benutzer mittels einer Grafischen Oberfläche (GUI) wurde JAVA™-*Swing* verwendet. Zum Betrieb ist Linux erforderlich.

2 Installation und sonstige Vorbereitungen

2.1 Installation

Unter der Voraussetzung, dass Sie eine den Anforderungen entsprechende Version von JAVA™ installiert haben gehen Sie bitte wie folgt vor.

1. Die Programm-Dateien von TSUNAMI finden Sie an folgender Stelle:

atzenger??:/irgendwo/nochweiter/?????

Kopieren Sie sich die Dateien in ein Unterverzeichnis (z. B. ~/tsunami/.) Ihres home-Verzeichnisses.

2. Kompilieren Sie die java-Dateien in dem von Ihnen gewählten Verzeichnis, analog zu folgendem Befehl:

javac –classpath /import/home/<your username>/tsunami
org/tum/zenger/tsunami/*.java

3. Starten Sie zum Test, ob bis hierhin alles funktioniert hat, das Programm analog zu folgendem Befehl:

java –classpath /import/home/<your username>/tsunami
org.tum.zenger.tsunami.Tsunami

Falls alles funktioniert, werden sie als erstes einen Splash-Screen sehen (Abb1):

Abb.1

2.2 ssh-Vorbereitungen

In der Einleitung wurde erwähnt, dass Sie die Möglichkeit haben, sich den Zielrechner für die Ausführung Ihres Testes selbst aussuchen können. Da die Netzwerkkommunikation für den Test über ssh bzw. scp abgewickelt wird und Sie vermutlich nicht andauernd Ihr Passwort eingeben wollen, wird dringend geraten, zur Public Key Authentisierung zwischen ihrem lokalen Rechner und dem Zielrechner über zu gehen, falls dies noch nicht gemacht ist.

Dies funktioniert folgendermassen:

1. Falls Sie in Ihrem ssh-Verzeichnis, normalerweise zu finden unter *~/.ssh*, noch keine Dateien mit folgenden Namen, *id_rsa.pub* und

id_rsa, finden, dann führen Sie als erstes in diesem Verzeichnis den Befehl

ssh-keygen –t rsa

aus. Nun passiert ungefähr folgendes

Generating public/private rsa key pair.
Enter file in which to save the key (~/.**ssh**/id_rsa): <ENTER>
Enter passphrase (empty for no passphrase): <ENTER>
Enter same passphrase again: <ENTER>
Your identification has been saved ~/.**ssh**/id_rsa.
Your public key has been saved in ~/.**ssh**/id_rsa.pub.
The key fingerprint is: aa:aa:aa:aa:aa: <your name>@localhost

2. Dann kopieren Sie den public key auf den entfernten Server und schreiben ihn in die Datei *authorized_keys*, so dass Sie sich auf diesen Server via public/private key Authentisierung einloggen können. Eine Möglichkeit dies zu machen ist:

   *~ > cat .ssh/id_dsa.pub | ssh <username>@<Zielrechner> *
   "umask 077; mkdir -p .ssh; cat >> .ssh/authorized_keys"

3. Zum Test sollten sie sich nun einmal per ssh auf dem Zielrechner einloggen, damit er in die Liste der *known_hosts* aufgenommen, da ansonsten während eines Testlaufs gefragt wird, ob Sie ihn aufnehmen wollen. Dies dient der Vermeidung von man-in-the-middle-attacks.

2.3 Optionale Vorbereitungen

Das Programm ermöglicht Ihnen verschiedene Compiler- und Linker-Optionen in einem Fenster für jeden Testfall auszuwählen. Diese flags und ihre default-Einstellungen sind in verschiedenen Property-files gespeichert und können den eigenen Wünschen angepasst werden (Siehe Programmierhandbuch Abschnitt 2.2 Compiler/Linker property Dateien).

3 Testen mit TSUNAMI – Anwendungsfall

3.1 Vorbereitungen

3.1.1 Das Makefile

Damit Sie die Eingaben, welche sie im Tsunami-*Settings*fenster (Abb.2)

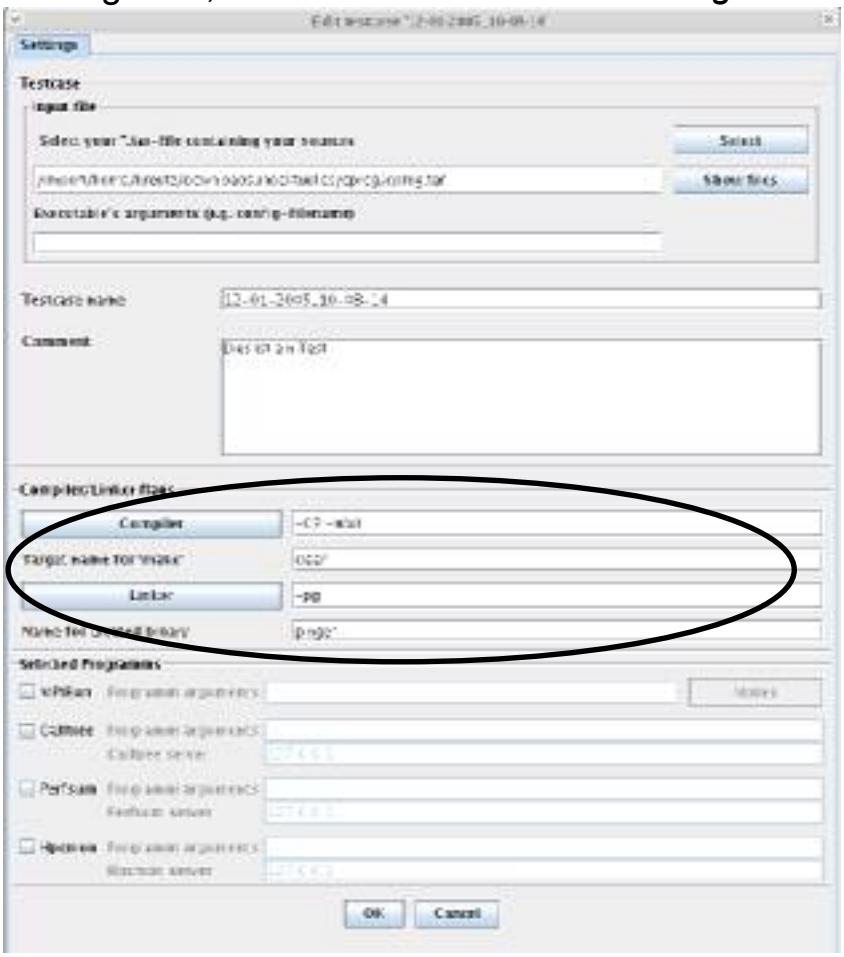


Abb.2

machen, auch nutzen können, müssen Sie Ihr *Makefile* folgendermassen anpassen.

1. Falls Sie wünschen unterschiedliche Compiler zu verwenden, zum Beispiel beim Rechnen auf dem Bode-Cluster auf den nodeX gcc und auf den Itanium-nodes icc, dann sollten sie überall dort, wo vorher der Compiler stand $(TSUNAMI_COMPILER) einfügen. Zum Beispiel:

   CC = $(TSUNAMI_COMPILER)

Damit wird die von TSUNAMI exportierte Umgebungsvariable ausgelesen.

2. Analog funktioniert dies für die Linker- und Compiler-flags. Beispielsweise *LDFLAGS += $(TSUNAMI_LINKERFLAGS) bzw. CFLAGS += $(TSUNAMI_COMPILERFLAGS)*.Hierbei ist es wichtig, das Plus nicht zu vergessen, damit fest eincodierte Flags nicht überschrieben werden, welche Sie in das Makefile eingebunden haben.

3. Der Name des Binaries muss durch *$(TSUNAMI_BINNAME)* ersetzt werden!

4. Das Verzeichnis für die auszuführende Datei muss *./bin* sein!

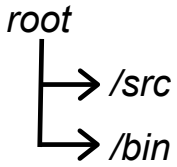
Beispielsweise

```
# Tsunami-Makefile fuer stream
# KM, 2004

CC      =       $(TSUNAMI_COMPILER)
LDFLAGS   +=      $(TSUNAMI_LINKERFLAGS)
CFLAGS  +=      $(TSUNAMI_COMPILERFLAGS)

LIBDIR          =       ./lib
BINDIR          =       ./bin


.SUFFIXES:
.SUFFIXES: .c .o

SRC   =       timer.c

default: $(BINDIR)/$(TSUNAMI_BINNAME)

all: clean $(BINDIR)/stream $(BINDIR)/tests docs

tests: $(BINDIR)/tests
```

```
$(BINDIR)/$(TSUNAMI_BINNAME) : $(SRC:%.c=$(LIBDIR)/%.o)
	$(CC) -c $(CFLAGS) -o $(LIBDIR)/main.o main.c
	$(CC) $(LDFLAGS) -o $@ $(SRC:%.c=$(LIBDIR)/%.o) $(LIBDIR)/main.o
$(LIBS)


$(BINDIR)/tests : $(SRC:%.c=$(LIBDIR)/%.o)
	$(CC) $(INCLUDES) -c $(CFLAGS) $(DEFS) -o $(LIBDIR)/tests/tests.o
tests/tests.c
	$(CC) $(LDFLAGS) -o $@ $(SRC:%.c=$(LIBDIR)/%.o) $(LIBDIR)/tests/tests.o
$(LIBS)


$(SRC:%.c=$(LIBDIR)/%.o) : $(LIBDIR)/%.o : %.c
	$(CC) $(INCLUDES) -c $(CFLAGS) $(DEFS) -o $@ $<

clean :
	$(RM) $(BINDIR)/$(TSUNAMI_BINNAME)
	$(RM) $(LIBDIR)/*.o
	$(RM) $(LIBDIR)/tests/*.o
```

3.1.2 Die .tar-Datei

Tsunami erwartet als Eingabe eine .tar-Datei mit folgender Struktur,



wobei das Makefile im *src*-Verzeichnis oder *root* liegen muss und das Binary muss nach dem Kompilieren im *bin*-Verzeichnis erzeugt werden!

Verpacken Sie alle nötigen Dateien – C-Dateien, Konfigurationsdateien usw. – entsprechend nach diesen Konventionen in ein Archiv.

3.2 Im Programm

3.2.1 Der Startbildschirm

3.2.1.1 Das „Test“-Fenster

Nach dem Splash-screen sehen sie das *Test*-Fenster (Abb. 3). Dies ist der Anfang und das Ende Ihrer Einstellungen. Von hier aus werden die Tests, welche sie aktiviert haben, mit dem *GO*-Knopf an der unteren Seite gestartet.

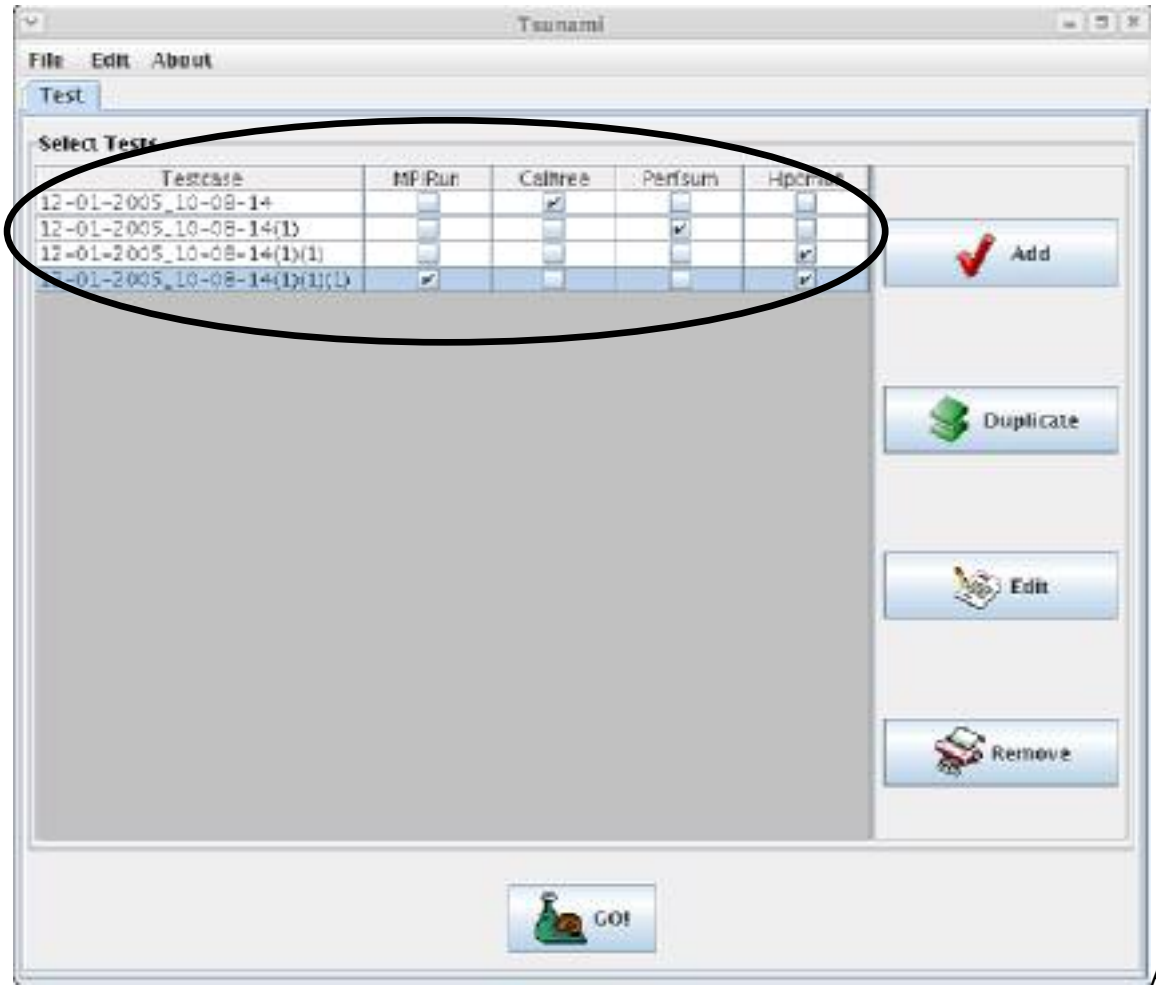


Abb.3

In dieser Abbildung sieht man bereits einige eingetragene Testfälle (innerhalb der Ellipse) im Feld *Select Tests*. Ganz links steht der *TestCase*-Name und daneben, aus den angekreuzten oder nicht

angekreuzten Schaltboxen, kann man herauslesen welches Testtool angewählt ist. Zu Beginn ist dieses Feld jedoch noch leer und sie müssen erst Testfälle erstellen.

Dazu gibt es folgende Funktionen:

- *Add*: Drücken Sie auf diesen Knopf, dann öffnet sich das *Settings*-Fenster (Abb. 2).
- *Duplicate*: Hiermit können Sie einen bereits vorhandenen Testfall kopieren. Zur Unterscheidung wird an den *Testcase*-Namen *(1)* angehängt.
- *Edit*: Bearbeiten Sie einen bereits vorhandenen und ausgewählten *Testcase*.
- *Remove*: Löschen Sie einen bereits vorhandenen und ausgewählten *Testcase*.

3.2.1.2 Das Menü

Diese Funktionen können auch über das Menü angewählt werden. Innerhalb des Menüpunktes *Edit* finden Sie alle oben genannten Funktionen wieder.
Dazu existieren auch noch folgende Tastenverknüpfungen

- STRG-N *Add*
- STRG-E *Edit*
- STRG-Einfg *Duplicate*
- STRG-Entf *Remove*

Innerhalb des Menüpunktes *Edit* existiert die Möglichkeit Property-Dateien sowohl für die Linker-Optionen (*Select linker property-file ...*) als auch für die Compiler-Optionen (*Select compiler property-file ...*) zu laden. Diese ermöglicht es Ihnen, zusätzlich zum Abspeichern der

Testfälle auch geeignete default-Belegungen abzuspeichern und wieder auszuwählen.

Im Menüpunkt *File* stehen folgende Auswahlmöglichkeiten zur Verfügung:

- *Load testcases ...* (STRG-L): Damit können sie abgespeicherte Zusammenstellungen von Testfällen wieder laden.
- *Save testcases ...* (STRG-S): Damit können Sie Zusammenstellung von Testfällen abspeichern.
- *Exit* (STRG-Q): Beenden des Programms.

Im Menüpunkt *About* bekommen sie mit dem Menüpunkt *About TSUNAMI* einige Informationen, wie zum Beispiel die Support-Adresse.

3.2.2 Das *Settings*-Fenster

Die genauen Einstellungen eines *Testcases* werden innerhalb des *Settings*-Fensters (Abb 2.), welches sich beim Neuanlegen eines *Testcases* oder beim Bearbeiten eines Alten öffnet.

Beim Anlegen eines Neuen sind nur folgende Felder vorbelegt:

- Textfeld *Testcase name* mit automatisch generiertem Namen (TT-MM-JJJJ_hh_mm_ss), welcher selbstverständlich geändert werden kann,
- Textfelder *Compiler flags* und *Linker flags* mit den im Property-File angegebenen default-Werten und
- die noch ausgeblendeten Textfelder <*Testtool*> *server*, welche mit *127.0.0.1 (localhost)* vorbelegt sind.

Einstellungen, bezogen auf das Fenster von oben nach unten:

1. Wählen Sie über den *Selec*t-Button das von Ihnen bereits erzeugte Archiv aus, der absolute Pfad erscheint dann im Textfeld leicht unterhalb und links.
2. Falls nötig, geben Sie Argumente für das auszuführende Programm mit. Zum Beispiel kann dies der Name einer Konfigurationsdatei sein, dieser sollte dann relativ zum *root* des Archivs adressiert werden.
3. Falls Sie noch Befehle auf dem Zielrechner ausführen wollen bevor der Kompiliervorgang startet, dann schreiben Sie dies in ein shell-Skript und übergeben Sie dieses per Auswahlfenster an das Programm. Dies kann nützlich sein, falls Sie zum Beispiel noch Bibliotheken zusätzlich einbinden müssen, und es hält das Programm äußerst flexibel
4. Ändern Sie, falls gewünscht den *Testcase name*.
5. Zur besseren späteren Identifizierung des Testes, geben Sie bei *Comment* Hinweise oder Klassifikation ein.
6. Öffnen Sie über den *Compiler*-Button das Auswahlfenster für die zusätzlichen zu den in Ihrem *Makefile* angegebenen Compiler-Optionen. Setzen Sie wie gewünscht Häkchen. Sollten Sie eine gewünschte Option nicht finden, dann können Sie sie per Hand im Textfeld eintragen.
7. *Target name for ‚make'*, falls default, dann ist keine Eingabe nötig.
8. Linker-Optionen analog zu 5. einstellen.
9. Wichtig! Geben Sie einen Namen für das erzeugte *Binary* an.
10. *Selected Programs*: Wählen Sie das gewünschte Testtool aus. Schreiben sie zusätzliche Argumente für dieses Testwerkzeug in die dazugehörigen Textfelder und geben Sie den Zielrechner an. Besonderheit bei MPIRun, über den Button *Nodes* können Sie die

gewünschten Knoten des Zielclusters auswählen (Abb. 5). Aus dieser Auswahl wird dann die Auswahl des geeigneten Compilers, zum Beispiel auf den Itanium-Knoten (itnodeXX) icc, getroffen. Desweiteren wird eine Konfigurationsdatei für das mpirun-Tool erzeugt, welche ihm die ausgewählten Knoten vorgibt.

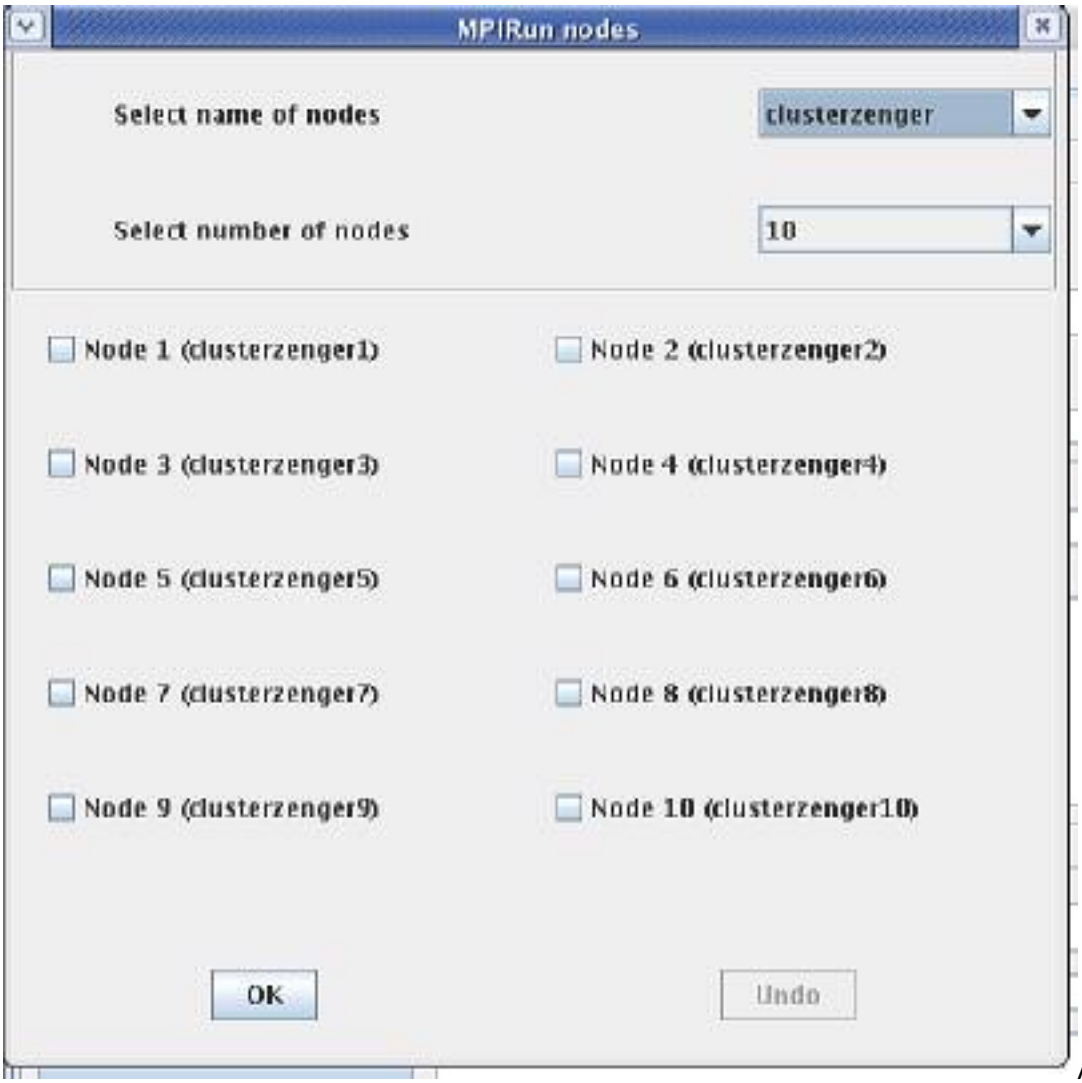


Abb. 5

11. Bestätigen Sie nun alles mit *OK*.

Sollten Sie nun alle gewünschten Testcases eingegeben haben, dann können Sie mit dem *GO*-Button im Test-Fenster diese Tests starten.

3.3 Testergebnis

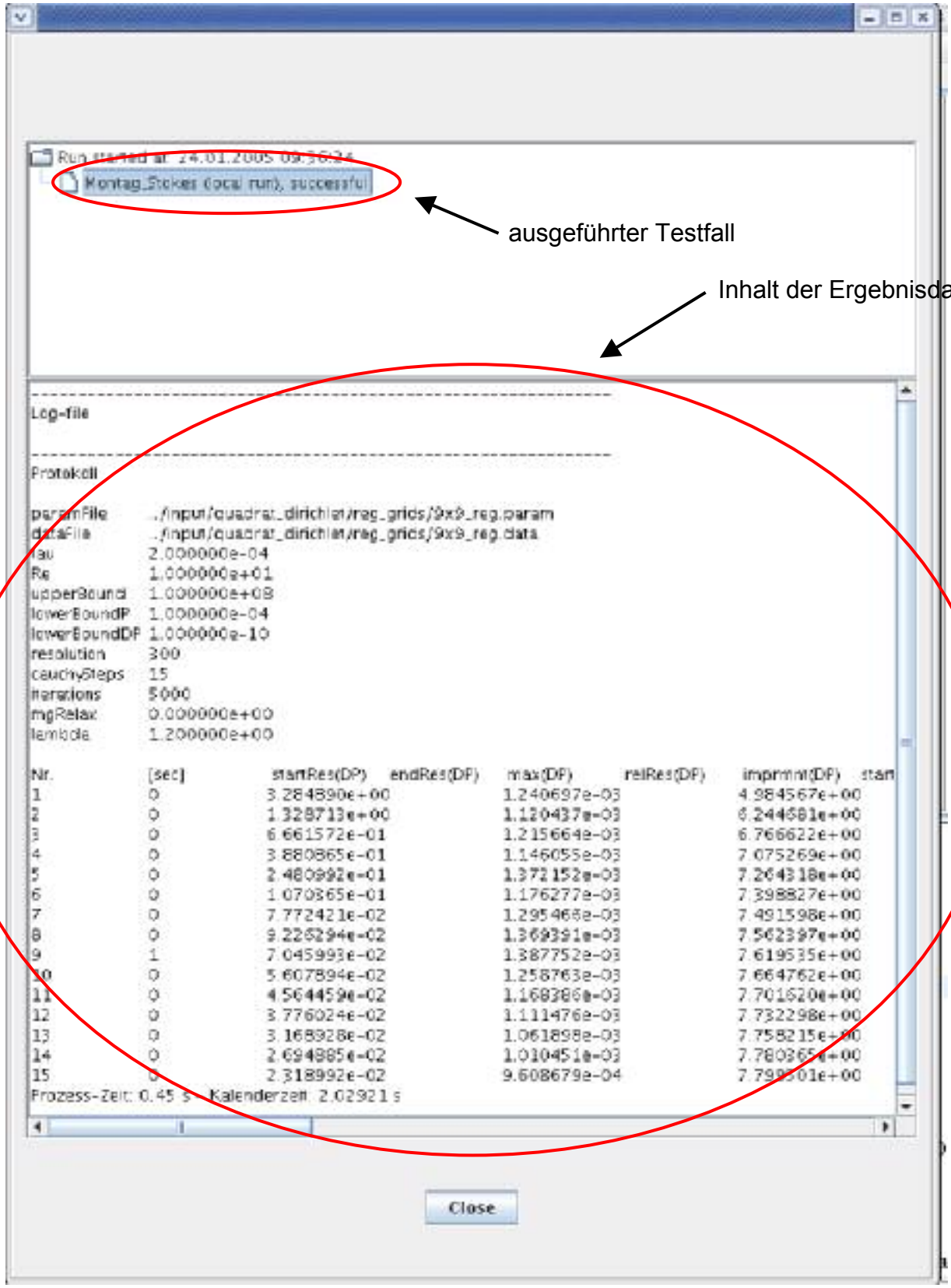
Sie erhalten nach dem Durchlauf der Tests in einem Dialog Auskunft über den Verlauf der Tests. Hierzu sind im oberen Teil des Dialogs alle ausgeführten Testfälle aufgelistet und auch ob diese erfolgreich (*run successful*) ausgeführt wurden. Dabei bedeutet ‚erfolgreich ausgeführt', dass die Ergebnis-Datei (*result_<testcase name>_<testtool>.log*) im Verzeichnis *~/tsunami_results/<testcase name>_<testtool>* existiert. In dieser Datei finden Sie die Standardfehlerausgabe und die Standardausgabe des Programmdurchlaufs Ihres kompilierten Programms. Weiterhin liegen in diesem Ordner zwei weitere Dateien

- *localError_<testcase name>_<testtool>.log*
- *remoteError_<testcase name>_<testtool>.log*

In diesen beiden Dateien stehen die möglicherweise aufgetretenen Fehler auf dem lokalen Rechner bzw. dem Zielrechner. Möglicher Inhalt der Zielrechnerfehlerdatei ist die Ausgabe des Compilers, weil Fehler beim Kompilieren aufgetreten sind. In diesem Fall würden Sie auch keine Ergebnisdatei in diesem Verzeichnis finden und es würde *run failed* dort stehen.

Wählt man nun einen Testfall aus, so wird im unteren Teil des Dialog der Inhalt der Ergebnis-Datei (im Falle einer erfolgreichen Ausführung) oder der Inhalt der Fehler-Dateien (im Falle einer fehlgeschlagenen Ausführung) angezeigt.

Siehe Abbildung auf der nächsten Seite.


Run started at 24.01.2005 09:56:34
Montag_Stokes (local run), successful
ausgeführter Testfall
Inhalt der Ergebnisdatei
Log-file
Protokoll
paramFile ./input/quadrat_dirichlet/reg_grids/9x9_reg.param
dataFile ./input/quadrat_dirichlet/reg_grids/9x9_reg.data
tau 2.000000e-04
Re 1.000000e+01
upperBound 1.000000e+08
lowerBoundP 1.000000e-04
lowerBoundDP 1.000000e-10
resolution 300
cauchySteps 15
iterations 5000
mgRelax 0.000000e+00
lambda 1.200000e+00
Nr. [sec] startRes(DP) endRes(DP) max(DP) relRes(DP) imprmnt(DP) start
1 0 3.284890e+00 1.240697e-03 4.984567e+00
2 0 1.328713e+00 1.120437e-03 6.244681e+00
3 0 6.661572e-01 1.215664e-03 6.766622e+00
4 0 3.880865e-01 1.146055e-03 7.075269e+00
5 0 2.480992e-01 1.372152e-03 7.264318e+00
6 0 1.670865e-01 1.176277e-03 7.398827e+00
7 0 7.772421e-02 1.295466e-03 7.491598e+00
8 0 9.226294e-02 1.369391e-03 7.562397e+00
9 1 7.045993e-02 1.387752e-03 7.619535e+00
10 0 5.607894e-02 1.258763e-03 7.664762e+00
11 0 4.564459e-02 1.168386e-03 7.701620e+00
12 0 3.776024e-02 1.111476e-03 7.732298e+00
13 0 3.168928e-02 1.061898e-03 7.758215e+00
14 0 2.694885e-02 1.010451e-03 7.780365e+00
15 0 2.318992e-02 9.608679e-04 7.799301e+00
Prozess-Zeit: 0.45 s Kalenderzeit: 2.02921 s
Close

4 Support

Falls Probleme beim Betrieb des Programmes auftreten sollten, dann können Sie sich an folgende Personen wenden:

- GUI: Nils Nitsch nitsch@in.tum.de
- Deployment: Martin Kreutz kreutz@in.tum.de