V*O*Rtech Computing

Experts in Technisch Rekenwerk

Postbus 260
2600 AG DELFT

tel. 015-285 0125
fax. 015-285 0126
vortech@vortech.nl

MEMO	EV/M05.040
Datum	15 juli 2005
Auteur(s)	Edwin Vollebregt
Onderwerp	Ontwikkeling van een MPI-versie van parallel WAQUA/TRIWAQ voor Myrinet

Documentinformatie

Versie	Auteur	Datum	Opmerkingen	Review
0.9	EV	15-07-2005	Concept rapportage van onderzoek.	BvtH
Bestandslokatie:		/v3/B04c_mpi_myrinet/report		

Samenvatting

In dit memo wordt verslag gedaan van werkzaamheden waarin de nieuwste versie van WAQUA/TRIWAQ in SIMONA, incl. nieuwe droogvalalgoritmes, geschikt is gemaakt voor gebruik van de MPICH-GM implementatie van MPI op het Linux-cluster van TUD-ct op basis van Myrinet. Hiervoor zijn de aanpassingen voor het nieuwe stopcriterium en voor het gebruik van MPI geïntegreerd in de nieuwste SIMONA-versie. Deze aanpassingen worden in dit memo uitgebreid gedocumenteerd en zijn gedegen getest met het oog op de door ons aanbevolen opname in de moederversie.

Er blijken slechts heel beperkte aanpassingen nodig te zijn voor het gebruik op een Linux-cluster op basis van Myrinet. En met de nieuwe versie wordt een uitstekende performance behaald. De hoogst gemeten speedup-factor is 21.7 bij gebruik van 28 processoren. Gegeven het onvermijdelijke effect dat de twee processoren per dual-processor PC elkaar in de weg zitten wanneer ze allebei aan het rekenen zijn is dit een praktisch ideale speedup.

1 Inleiding

In het project "Sterkte en Belasting Waterkeringen" worden simulaties gebruikt om inzicht te krijgen in de doordringing van deiningsgolven in de Waddenzee. Enkele principesommen hebben aangetoond dat hiervoor een variant van TRIWAQ met uitbreidingen voor niet-hydrostatisch rekenen moet worden gebruikt voor een fijnmazig rooster. Schattingen geven aan dat de doorlooptijd per som zou kunnen oplopen tot 30 dagen bij gebruik van een PC met 2.0 GHz Pentium IV processor.

Om de doorlooptijd van de berekeningen te kunnen terugdringen wil Rijkswaterstaat/-RIKZ gebruik maken van het Linux-cluster van de sectie Vloeistofmechanica van Civiele Techniek van de TU Delft, kortweg CT-cluster genoemd. Dit systeem bevat 32 AMD Athlon

1800+ processoren, waarmee de rekentijd in het meest ideale geval tot minder dan 1 dag per som kan worden teruggebracht.

Een probleem is wel dat het communicatiesysteem MPICH-GM dat op het CT-cluster beschikbaar is niet direct door parallel WAQUA/TRIWAQ wordt ondersteund. MPICH-GM implementeert de standaard MPI versie 1, terwijl de MPI-versie van WAQUA/TRIWAQ op versie 2 van de MPI-standaard is gebaseerd. Toch is het gebruik van MPICH-GM nodig om de gewenste versnelling van de berekeningen te bereiken. Daarom moet er wat programmeerwerk en testen op het CT-cluster worden uitgevoerd.

Een ander aspect is dat de MPI-versie van parallel WAQUA/TRIWAQ die V∅Rtech Computing in 2004 in opdracht van WLH Borgerhout heeft ontwikkeld op een iets oudere SIMONA-versie is gebaseerd. In 2004 is er in het kader van het zogenaamde "droogvalproject" ook een grote verbetering van de droogval- en onderloopprocedures in WAQUA/TRIWAQ gerealiseerd. De resultaten van dat project zijn in juni 2005 in de moederversie van WAQUA/TRIWAQ in SIMONA opgenomen. De ervaringen hiermee zijn dat deze nieuwe algoritmes de stabiliteit en robuustheid van berekeningen met WAQUA/TRIWAQ sterk verbeteren.

In juni 2005 heeft Rijkswaterstaat/RIKZ bij V∅Rtech offerte aangevraagd en daarna opdracht verstrekt voor de volgende werkzaamheden:

- Integreren van het zogenaamde "nieuwe stopcriterium" en van de aanpassingen ten behoeve van het gebruik van MPI in de nieuwste moederversie van WAQUA/TRIWAQ in SIMONA gebaseerd op de algoritmes van het droogvalproject.
- Onderzoeken van de beperkingen van de MPI-versie voor Myrinet die op het CT-cluster wordt gebruikt, bepalen van de benodigde aanpassingen aan parallel WAQUA/TRIWAQ en implementatie hiervan.
- Installeren en testen van de aangepaste programmatuur op het CT-cluster.
- Tenslotte wordt er rapportage over de werkzaamheden en de performance gemaakt.

De integratie van de benodigde uitbreidingen van TRIWAQ voor niet-hydrostatisch rekenen vallen buiten deze opdracht.

Dit memo betreft de rapportage over deze opdracht. Hierbij worden de verschillende activiteiten in aparte paragrafen besproken. Hierbij wordt extra aandacht besteed aan het beschrijven van de aanpassingen voor het nieuwe stopcriterium en voor MPI, met het oog op de gewenste opname van de wijzigingen in beheer en onderhoud. Voor niet-technisch geïnteresseerden is vooral paragraaf 4 over de uitgevoerde performance-experimenten van belang.

2 Integratie van uitbreidingen in de moederversie van WAQUA/TRIWAQ

Het nieuwe stopcriterium en de overstap van communicatiesysteem PVM naar MPI zijn beiden bedoeld om de performance te verbeteren van parallelle runs. De eerste aanpassing is begin 2004 ontwikkeld in opdracht van RIZA [2]. De tweede aanpassing is in het tweede kwartaal van 2004 gerealiseerd in opdracht van het Waterbouwkundig Laboratorium Borgerhout [1].

Beide uitbreidingen zijn in eerste instantie gerealiseerd voordat de DDHOR+VERT versie van WAQUA/TRIWAQ was geoperationaliseerd. In november 2004 zijn ze beiden in de nieuwe export-versie van SIMONA (2004-01) geïntegreerd.

In het huidige project zijn we uitgegaan van de moederversie van SIMONA van begin juni 2005, waarin zojuist fase 3 van het droogvalproject is opgenomen. Daarnaast hebben we de versies `simona0410-test` en `simona0410-test-mpi` gebruikt, waarin de uitbreidingen in de exportversie 2004-01 zijn geïntegreerd.

De integratie is in twee aparte stappen gedaan. Allereerst is er een versie `simona0506-test` gerealiseerd waarin alle relevante uitbreidingen aan WAQUA/TRIWAQ die niets met MPI te maken hebben zijn overgenomen. Daarna is deze nieuwe versie gekopieerd naar `simona0506-test-mpi` en zijn hierin de aanpassingen voor MPI geïmplementeerd. Deze tweedeling maakt het mogelijk om de twee aanpassingen apart van elkaar te beoordelen en op te nemen in de moederversie van de programmatuur.

2.1 Integratie van het nieuwe stopcriterium

De grootste verschillen tussen de versie met het nieuwe stopcriterium en de oorspronkelijke versie van de programmatuur betreffen natuurlijk het nieuwe stopcriterium zelf:

- Het stopcriterium zelf is geïmplementeerd in subroutine `wasck2`. Verder zijn er aanpassingen voor gemaakt in subroutines `waspif`, `wastgd`, `waschk`, `wasspu/v`, `wassuc`, `wastru/v`, `wasdfc` en `waspnd`.

- Van subroutine `wasck2` is er ook een tweede versie gemaakt, `wasck2.0506.f`, die met dezelfde subroutineheader het oude stopcriterium implementeert. Hiermee kan gemakkelijk tussen de twee criteria worden geschakeld, bijvoorbeeld om bij het onderzoeken van problemen te kijken wat het effect van het stopcriterium is.

- Bij de installatie van versie `simona0410-test` voor RIZA is gebleken dat aanpassingen aan het nieuwe stopcriterium ten behoeve van domein decompositie leidden tot een significante overhead. Daarbij zijn ook mogelijke verfijningen bedacht om de overhead weer teniet te doen. Een van die verfijningen is in de nieuwe versie geïmplementeerd.

  Het betreft de range waarvoor het veldarray `iwfld` wordt gevuld. In plaats van dit voor het hele subdomein te doen wordt het nu alleen bij interfacepunten gedaan.

- In subroutines `waspif` en `wasrnm` is de grootte en bepaling van het array GLOBROWNUM uitgebreid. Er is een tweede kolom toegevoegd waarin het volgnummer van een subrij binnen de corresponderende globale rij wordt opgeslagen. Deze informatie is van belang bij het debuggen van situaties met betrekking tot het nieuwe stopcriterium.

- In subroutines `wasdfc`, `wasdff` en `waspnd` is een probleem gerepareerd dat met het nieuwe stopcriterium te maken heeft. In plaats van dat array `isbbou` in een keer voor alle rijen en kolommen samen wordt doorgegeven worden er nu aparte argumenten `isbbou` voor de rijen en `isbcol` voor de kolommen gebruikt. Het probleem trad op bij spiegeling van de rijen en kolommen wanneer het nieuwe criterium werd gebruikt.

- In subroutines `wastrd` en `waspnd` is een correctie doorgevoerd in het nieuwe stopcriterium ten opzichte van versie `simona0410-test`. In sommige situaties werd de randafhandeling niet correct gedaan. Het effect hiervan is onduidelijk. Waarschijnlijk leidt de fout tot verschillen op afbreekniveau van het iteratieproces.

Daarnaast zijn er ook een aantal kleinere correcties en aanpassingen doorgevoerd:

- In COCLIB is een trucje overgenomen waarmee kan worden gecontroleerd of het argument ICOCAD van veel subroutines valide is. Deze controle is echter nog niet geactiveerd.

- In COCLIB is in subroutine `cocstt` een kleine fout gerepareerd. Hierbij zagen we dat men in de behandeling van melding P03019 veel write-statements door calls van `siinfm` (informational message) vervangen heeft. Dit is volgens ons voor subroutine `cocstt` ongewenst, omdat de print-uitvoer hierdoor minder overzichtelijk wordt.

- In COPPRE is een uitbreiding in de afhandeling van enclosures in areas-files geïmplementeerd. Deze uitbreiding heet nog "experimenteel", maar functioneert al een tijd zonder dat er problemen mee zijn gemeld. Deze uitbreiding is nodig om bepaalde areas-files die met VISIPART zijn gegenereerd te kunnen gebruiken in parallelle runs.

- In WAQPRO is een correctie overgenomen in het transportgedeelte met betrekking tot "wrap-around".

- In WAQPRO zijn in subroutine `waschk` de argumenten `mypart` en `npart` toegevoegd. Deze zijn in een integratieslag door MX.*Systems* weggehaald omdat ze niet worden gebruikt. Maar bij het debuggen van problemen kunnen de argumenten wel degelijk nodig zijn.

  De aanpassing heeft gevolgen voor een aantal rekenroutines, omdat `mypart` en `npart` moeten worden doorgegeven tot op het niveau waar ze worden gebruikt: `wasspu/v`, `trscue`, `trsjam`, `trssuw`, `wasuxc`, `wassuc`, `wastru/v`, `trsdif`, `trsjac`, `trstur`, `trsdgs`.

- In subroutine `waspst` is het format voor het printen van cpu-tijden aangepast. Er wordt iets minder detail gegeven (resolutie van 1 $msec$ is voldoende), en tegelijkertijd worden er grotere tijden ondersteund.

De aanpassingen zijn getest met behulp van de SIMONA testbank (aangepaste versie van V∅RTech). Hierin zitten een honderdtal simulaties voor allerlei scenario's met WAQUA en TRIWAQ.

De simulaties met TRIWAQ laten allemaal nul verschillen zien. Dat is tot op zekere hoogte logisch omdat er in TRIWAQ nauwelijks aanpassingen zijn gemaakt.

In simulaties met WAQUA zijn in een groot aantal modellen kleine verschillen te zien. Deze verdwijnen voor vrijwel alle sequentiele runs wanneer compileroptimalisaties worden uitgezet.

In tien simulaties blijven er toch verschillen bestaan (`bak.f.ba45.trans`, `bak.f.bast`, `grev`, `ijssel`, `maas`, `maas1`, `maasdemo`, `n10`, `waal_viscrefterms_on`). Het blijkt dat in al deze gevallen de "cycle-check" van subroutine `waschk` tenminste een keer actief is geweest. In het oude stopcriterium werd `waschk` niet gebruikt voor het iteratieproces voor de waterstanden in subroutine `wassuc`. In het nieuwe stopcriterium wordt ze wel gebruikt, en wordt het iteratieproces voortijdig afgebroken wanneer er in drie achtereenvolgende iteraties geen vooruitgang wordt geboekt.

De verschillen verdwijnen wanneer de cycle-check in de nieuwe versie tijdelijk wordt uitgezet. In sequentiele runs zijn er dan alleen nog verschillen in de modellen `maas` en `maas1`. Dit blijkt te komen door een wijziging uit het droogvalproject. In deze modellen blijkt er soms in de 20^e iteratie nog droogvallen plaats te vinden. In de versie met het nieuwe stopcriterium werd er dan toch een 21^e iterate uitgevoerd, in de oude versie niet. Dit laatst gedrag is overgenomen omdat dit bij de nieuwe algoritmes voor droogvallen en onderlopen hoort.

In parallelle en domeindecompositieruns ontstaan er ook steeds kleine verschillen. De wat grotere verschillen in de simulaties `ddh_test_3b`, `ddh_test_9` en `ijsm_parallel` blijken met de cycle-check samen te hangen. Voor het overige zijn de verschillen het gevolg van compileroptimalisaties, veranderingen in het zogenaamde "minit-schema" en van het nieuwe stopcriterium.

2.2 Integratie van de aanpassingen t.b.v. MPI

De aanpassingen die er aan parallel WAQUA/TRIWAQ moeten worden gemaakt voor de overstap van PVM naar MPI zijn vrijwel volledig in COCLIB en COEXEC gelokaliseerd. In deze deelsystemen zijn vrij omvangrijke aanpassingen gemaakt:

- COEXEC: 9 routines aangepast, 1 nieuw, circa 700 regels verschil.
- COCLIB: 20 routines aangepast, 2 nieuw, 1 verwijderd, circa 2.000 regels verschil.

Voor het overige zijn de benodigde aanpassingen als volgt:

- Omdat het opstarten van MPI anders gaat dan van PVM moet het bestand `hostfile.-gen` worden aangepast.
- In de runprocedure `waqpro.run` zijn diverse aanpassingen nodig. Onder andere in het hercompileren van executables met een afwijkende buffersize. Daarnaast voor het opstarten van sequentiele en parallelle runs. Tenslotte moet in parallelle runs de uitvoer per rekenproces na afloop van de run worden samengevoegd.
- In het hoofdprogramma van WAQPRO wordt in parallelle runs de schermuitvoer omgeleid naar een tijdelijke uitvoerfile.

De MPI-versie `simona0506-test-mpi` is ten eerste getest door een paar sequentiële sommen van de testbank te draaien. Deze lieten geen van allen verschillen ten opzichte van de uitgangsversie `simona0506-test` zien.

Vervolgens zijn alle parallelle runs en domeindecompositieruns van de testbank uitgevoerd. Nadat alle deelsystemen met dezelfde compileropties waren gecompileerd waren hierin geen verschillen ten opzichte van de uitgangssituatie te zien.

3 Aanpassingen aan parallel WAQUA/TRIWAQ voor Myrinet

Door middel van testen met eenvoudige testprogramma's zijn de beperkingen van de MPI-implementatie op het CT-cluster achterhaald. Vantevoren werden de volgende beperkingen verwacht:

- De MPI-implementatie voor Myrinet "MPICH-GM" is gebaseerd op de vrij verkrijgbare MPI-implementatie MPICH. Een beperking van deze implementatie is dat hij geen dynamisch opstarten van taken ondersteunt ("`MPI_Spawn`"). Dit is geen groot probleem voor parallel WAQUA/TRIWAQ omdat deze functie niet (meer) wordt gebruikt. Het dynamisch opstarten van taken door het masterproces COEXEC leverde ook bij gebruik van de MPI-implementatie LAM-MPI versie 7.0 problemen op.

- De run-procedure `waqpro.run` van parallel WAQUA/TRIWAQ gebruikt het commando `mpiexec` om de parallelle processen op te starten. Een beperking van MPICH-GM is echter dat er in berekeningen die met dit commando worden opgestart geen gebruik kan worden gemaakt van de Myrinet interface. Dit vergt kleine aanpassingen aan de run-procedure, waarin `mpirun` moet worden gebruikt in plaats van `mpiexec`.

- Een beperking van de MPICH-GM implementatie van MPI lijkt te zijn dat er per berekening slechts 1 executable kan worden opgestart. Dit zou grote consequenties hebben voor parallel WAQUA/TRIWAQ omdat er dan geen apart masterproces COEXEC kan worden gebruikt. COEXEC voert echter wel de nodige taken uit, bijvoorbeeld het controleren van de invoer voor domeindecompositieruns. Deze taken zouden dan op een andere manier in de runprocedure moeten worden verwerkt.

Met name met betrekking tot het derde punt bleek de situatie gelukkig mee te vallen. Er kunnen toch meerdere executables worden gebruikt door gebruik te maken van de `procgroup`-optie van `mpirun`.

Op het Linux-cluster van TUD-ct moeten alle parallelle runs via het wachtrijsysteem PBS worden opgestart. Daarom is het genereren van de `procgroup`-file alleen voor deze situatie uitgewerkt. Een typisch voorbeeld van zo'n file is als volgt:

```
#
# --- hostfile21223 - hostfile for Parallel WAQUA/TRIWAQ.
#     generated by waqpro.run using NODES-file of OpenPBS
#
#     list of hosts for the configuration:
#
node12 0 /home/rws1/vortech/simona0506-test-mpi/bin/coexec.exe
node12 1 /home/rws1/vortech/simona0506-test-mpi/bin/waqpro.exe
```

```
node13 1 /home/rws1/vortech/simona0506-test-mpi/bin/waqpro.exe
node14 1 /home/rws1/vortech/simona0506-test-mpi/bin/waqpro.exe
node15 1 /home/rws1/vortech/simona0506-test-mpi/bin/waqpro.exe
```

Zodra deze file is gegenereerd wordt de daadwerkelijke run opgestart met het commando

```
mpirun -pg hostfile$$ `which $exe_exc` < control$$
```

Hierin geeft de optie `-pg hostfile$$` aan welke executables er op welke nodes moeten worden opgestart. Om de een of andere reden moet er daarna alsnog een executable worden opgegeven aan het `mpirun`-commando. Dit wordt gedaan met het argument `` `which $exe_exc` ``, dat expandeert tot `/home/rws1/vortech/simona0506-test-mpi/bin/coexec.exe`. Dit is de executable waarnaartoe de standaard invoer stroom (`< control$$`) wordt doorgegeven. Vervolgens moet dezelfde executable ook nog in de procgroup-file worden genoemd, met *count* `0`. De precieze achtergrond hiervan is onduidelijk, maar testen geven aan dat deze manier van werken goed voldoet.

Een *feature* van de MPI-versie van parallel WAQUA/TRIWAQ is dat ook sequentiële runs op de MPI-manier moeten worden opgestart. In dit geval laten we geen procgroup-file genereren. Het commando dat wordt gebruikt is

```
mpirun -machinefile $PBS_NODEFILE -np 1 `which $exe_pro` < control$$
```

Hierin staat dat de executable `` `which $exe_pro` `` (`waqpro.exe`) een keer (`-np 1`) moet worden op gestart op de eerste van de machines die in `$PBS_NODEFILE` worden genoemd.

De laatste aanpassing die tenslotte aan de MPI-versie van WAQUA/TRIWAQ moest worden gemaakt voor het gebruik op het CT-cluster betreft het hercompileren van executables met een afwijkende buffersize. Hierin moeten andere bibliotheken worden meegelinkt voor MPICH-GM dan voor de LAM-implementatie van MPI:

```
     if [ $UI_NAME = linux ]; then
#        MPILIBS="-L$MPI_ROOT/lib -pthread -llammpio -llamf77mpi -lmpi \
#                 -llam -lutil"
         GM_ROOT=/opt/gm
         MPILIBS="-L$MPI_ROOT/lib -lmpichf90 -lmpichfarg -lmpich \
                  -L$GM_ROOT/lib -lgm -lpthread"
      fi
```

De uitgecommentarieerde regels zijn voor LAM-MPI, de andere regels voor MPICH-GM. Op termijn zou de gebruikte MPI-implementatie instelbaar kunnen worden gemaakt. Bijvoorbeeld op de manier waarop dit is gedaan met de Fortran-compiler voor het Linux-platform:

```
   elif [ $UI_NAME = linux -a "$LINUX_F77" = intel ]; then
      ifort ...
   elif [ $UI_NAME = linux -a "$LINUX_F77" = GNU ]; then
      g77 ...
```

4 Installatie en testen van de nieuwe versie van de programmatuur

De installatie van de aangepaste SIMONA-versie is zonder noemenswaardige problemen gegaan. Het enige is dat het werk ruime tijd heeft stilgelegen omdat MPI op het cluster niet correct functioneerde.

De correctheid van de installatie is op de voor ons gebruikelijke manier getest. Hierbij lopen we een checklist af die allerlei moeilijke punten voor de installatie bevat: printen van user-id's, runtijden, verwerking van het TAB-karakter, en gebruik van afwijkende buffersizes in allerlei verschillende omstandigheden. Ook wordt kort gekeken naar de uitvoer van het testmodel op drie debietraaien. Hierin zijn geen afwijkingen geconstateerd.

4.1 Performance van sequentiële runs voor het CSM8-model

De performance van de nieuwe versie is in eerste instantie met een versie van CSM8 voor TRIWAQ onderzocht. In deze versie worden 5 lagen gebruikt en worden 4 dagen werkelijke tijd gesimuleerd.

De eerste test betreft verschillende sequentiële runs. Sequentiële runs voor dit model op een verder onbelast systeem kosten 427.0 *sec* (gemiddelde over drie runs, spreiding $\pm 1.5\,sec$). Dit is 1.6 keer langzamer als wat we recent op nieuwe Linux-PC's van HKV lijn in water hebben gemeten.

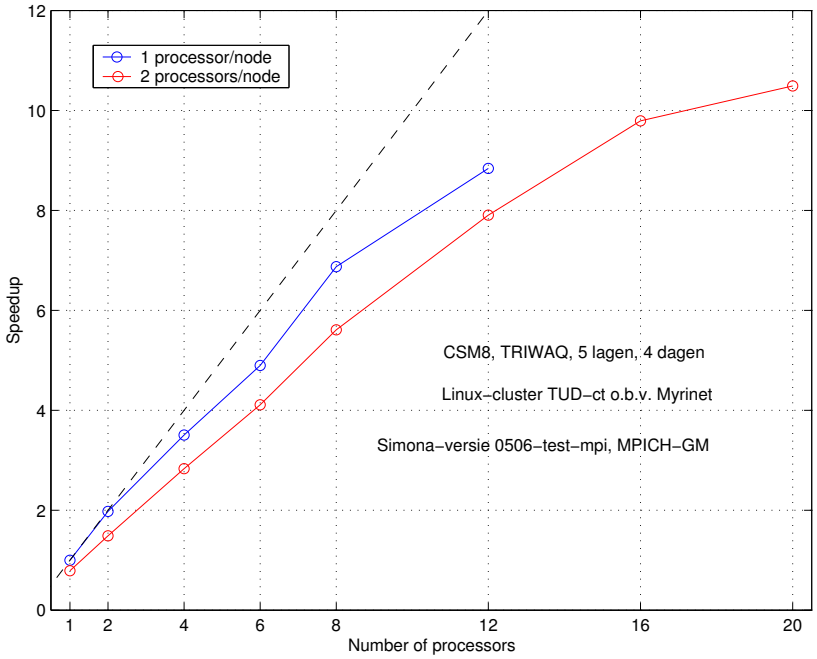
Wanneer er twee sequentiële runs tegelijkertijd op dezelfde node van het cluster worden gedraaid dan beïnvloedt dit de performance negatief. Dit noemen wij het *"dual-processor effect"*. De gemiddelde doorlooptijd wordt dan 539.7 *sec* (gemiddelde van 9 runs, range $524 - 570\,sec$). De rekensnelheid per processor is dan nog zo'n 79% van de oorspronkelijke snelheid wanneer er slechts 1 processor wordt gebruikt.

In de experimenten hierboven werd er geen MAP-uitvoer geproduceerd. Op die manier wordt de pure rekensnelheid onderzocht. In een andere variant van CSM8 laten we extreem veel output produceren. In iedere tijdstap worden de velden op de SDS-file gezet. De simulatieduur wordt hierbij tot 1 dag teruggebracht. De totale doorlooptijd wordt dan 150.7 *sec*. Van deze tijd wordt er 52 *sec* besteed aan het schrijven van velden. De uitendelijke SDS-file wordt $534.5\,MB$ groot. De schrijfsnelheid van WAQPRO is dus circa $10.3\,MB/sec$. Dit is zeker niet hoog, maar echte beperkingen hoeven hier niet van te worden verwacht.

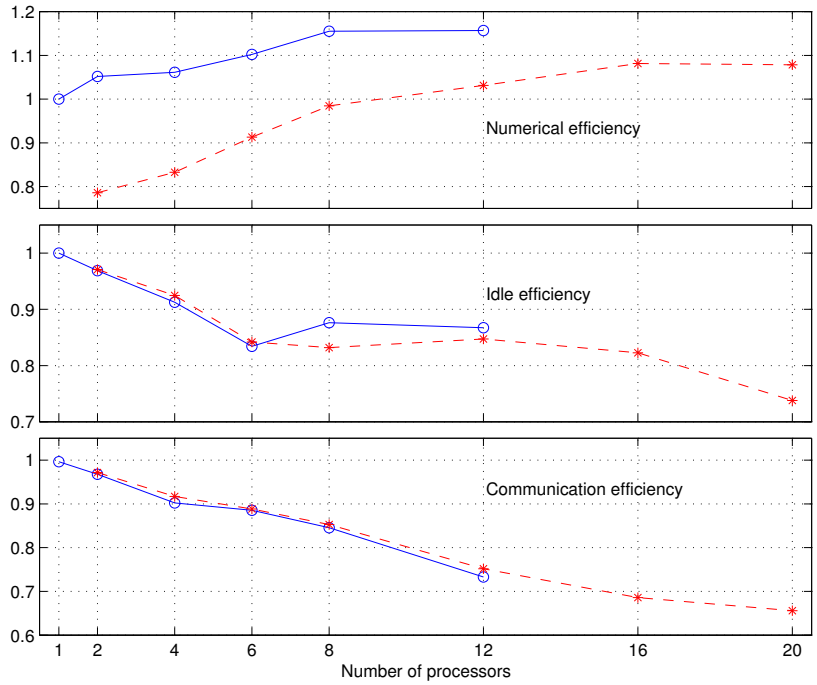
4.2 Parallelle performance voor het CSM8-model

De parallelle performance voor het CSM8-model met TRIWAQ wordt getoond in Figuur 1. De figuur laat zien dat er een goede versnelling wordt bereikt. Bijvoorbeeld wordt er op Linux-clusters op basis van Gigabit Ethernet op 8 processoren vaak een speedup van circa 5.5 behaald. Dit verschil komt doordat de communicatietijd is verkleind, zowel door het gebruik van MPI als van Myrinet. Daarnaast zijn de processoren ook relatief iets minder snel, waardoor de communicatietijd relatief nog minder belangrijk wordt.

De rode lijn betreft sommen waarbij beide processoren per node van het cluster worden benut en hebben daardoor last van het dual processor effect. Wanneer hiervoor wordt gecorrigeerd dan wordt er zelfs op 20 processoren nog een uitstekende speedup behaald. Wanneer



Figuur 1: *Speedup voor het CSM8-model met TRIWAQ (5 lagen), apart voor runs waarbij 1 processor per node en 2 processors per node worden benut.*



Figuur 2: *Uitsplitsing van de efficientie van de parallelle berekening voor het CSM8-model met TRIWAQ (5 lagen). Van boven naar beneden worden de relatieve rekensnelheid, de mate van load balancing, en de communicatie overhead getoond.*

je namelijk 20 sequentiële sommen tegelijk zou uitvoeren dan haalde je een performance van $0.79 \cdot 20 = 15.8$, een speedup van 10.5 betekent dus een efficientie van ruim 66%.

Via bepaalde definities kan de efficientie worden gesplitst in

$$E = E_{num} \cdot E_{idle} \cdot E_{comm} \tag{1}$$

De drie factoren die hierin worden onderscheiden zijn

E_{num} numerieke efficientie: hoeveel rekenwerk wordt er in een parallelle som gedaan, en hoe snel gaan de berekeningen t.o.v. een sequentiele som (o.a. cache effecten).

E_{idle} load balancing efficientie: hoe goed (gelijkmatig) is het rekenwerk over de processoren verdeeld.

E_{comm} communicatie efficientie: hoeveel moet er worden gecommuniceerd en hoe snel gaat die communicatie.

Figuur 2 laat de resultaten van deze uitsplitsing zien voor het CSM8-model. Hierin zijn de blauwe lijnen weer voor simulaties met gebruik van één processor per node en de rode lijnen voor gebruik van twee processors per node.

De bovenste grafiek in de figuur toont de verandering van de effectieve rekensnelheid. In beide gevallen neemt de rekensnelheid toe naarmate er meer subdomeinen worden gebruikt. Dit is het cache voordeel dat vaak in parallelle berekeningen wordt gezien. Alleen start de rode lijn op een waarde onder de 0.8, ten gevolge van het dual processor effect. De toename in de rode lijn is een stuk groter dan in de blauwe. Dit is ook een bekend effect; naarmate subdomeinen langer moeten wachten op communicaties met elkaar (grotere communicatie overhead en/of load imbalance) hebben ze ook langer de processor geheel voor zichzelf. Het dual processor effect speelt dan een minder grote rol. Dus communicatie en load imbalance worden voor een gedeelte gemaskeerd door afname van het dual processor effect.

De middelste grafiek in Figuur 2 laat zien dat er een behoorlijk grote load imbalance is opgetreden. In de uitgevoerde experimenten hebben we simpelweg de default automatische partitiemethode gebruikt. Met handmatige tuning van de opdeling moet dus nog een stuk hogere speedup kunnen worden bereikt. Alleen speelt de Intel-compiler hierin een vervelende rol; in experimenten voor WLH Borgerhout is gebleken dat de rekentijd per subdomein vrij grillig afhangt van de vorm van het subdomein. De achtergronden hiervan zijn nog onbekend en zouden voor het optimaliseren van de performance eens moeten worden uitgezocht.

De onderste grafiek toont tenslotte een inschatting van de communicatieoverhead. Deze lijnen zijn conform onze verwachtingen. Alleen is opvallend dat de rode lijn nog net iets boven de blauwe ligt. Onze verwachting was dat de communicatieoverhead zou toenemen omdat er steeds twee processen gebruik maken van dezelfde link van het Myrinet. Het effect treedt echter ook op in het testmodel “bak_parll”, zie Figuur 4 in paragraaf 4.3. De oorzaak van dit effect zit misschien in de communicatie tussen processen op dezelfde node. Deze verloopt sneller dan de communicatie tussen verschillende nodes omdat er helemaal geen Myrinet-link bij betrokken is. Anders speelt misschien de afname van de rekensnelheid door het dual processor effect een rol, waardoor de communicatie-tijd met een grotere rekentijd vergeleken wordt en dus relatief kleiner wordt.

4.3 Parallelle performance voor het model "bak_parll"

Het CSM8-model is met zijn 173×201 punten en vullingsgraad van 58% eigenlijk te klein om de parallelle performance te onderzoeken op een groot aantal processoren. Specifiek voor dit doel hebben we daarom een eenvoudig testmodel opgezet dat "bak_parll" is genoemd.

Het model "bak_parll" schematiseert een estuarium van $18 \times 18\,km$. Alleen de linker zijde is open. Op deze rand wordt de waterstand voorgeschreven met een verhang van $40\,cm$ tussen het onderste en bovenste punt, waardoor er een stroming op gang wordt gebracht. De diepte is constant $10\,m$. Er wordt met zout en $k-\epsilon$ turbulentie gerekend. De initiële zoutverdeling wordt opgegeven met cornervalues respectievelijk $35.0, 18.0, 6.0$ en $25.0\,kg/m^3$ tegen de klok in, startend in de linker onderhoek. Tenslotte ligt er een bronpunt rechtsonder waar continu $1.500\,m^3/s$ zoet water wordt geloosd. Deze gegevens zijn allemaal vrij arbitrair gekozen met als belangrijkste doel dat er een normaal aantal iteraties wordt uitgevoerd.

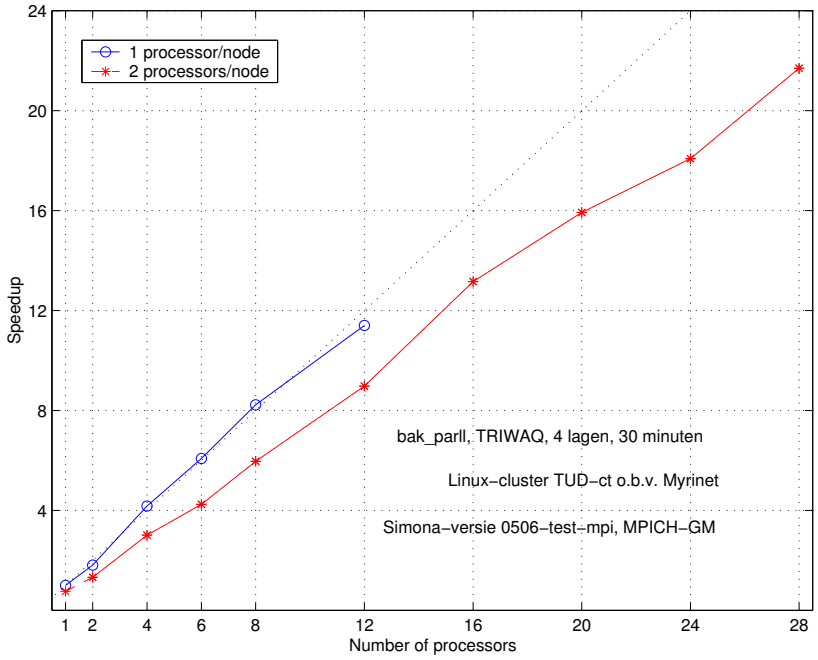
Het aantal roosterpunten dat wordt gebruikt is $362 \times 362 \times 4$. Dit is zodanig gekozen dat het rooster exclusief randpunten in beide richtingen gemakkelijk verdeeld kan worden over 2, 3, 4, 5, 6 en 8 processoren. Het is niet onderzocht of het gebruiken van 360 punten horizontaal voor de parallellisatie voordelig is. De roosterafstand is $50\,m$. Er wordt een tijdstap van $30\,sec$ gebruikt. Met de lengte van de simulatie kan worden gespeeld; op het huidige cluster levert een simulatie van 30 minuten een doorlooptijd van sequentiële runs van $401.4\,sec$ op. Dit is lang genoeg voor enige nauwkeurigheid van de metingen, en kort genoeg dat testen vlot kunnen worden uitgevoerd.

De genoemde tijd is voor een sequentiële run waarbij er slechts één processor per node is bezet. Wanneer er twee sequentiële runs voor dit model tegelijk worden uitgevoerd dan is de doorlooptijd $524.8\,sec$. Dit is weer het gemiddelde over 9 runs. De metingen liggen in de range $518 - 542\,sec$. Voor dit model komen we hiermee op een dual processor effect van 76.5%.

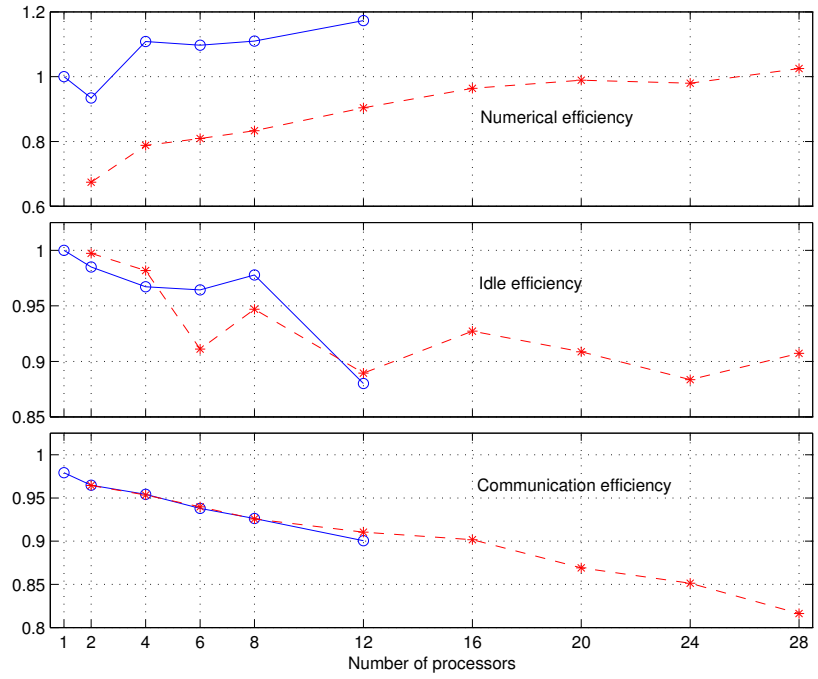
De parallelle performance voor dit model wordt getoond in de Figuren 3 en 4. Deze figuren laten zien dat er op het CT-cluster voor grotere TRIWAQ-modellen een uitstekende speedup en goede schaalbaarheid wordt behaald. Beide speedup-curves verlopen vrijwel lineair, er zijn nog geen sporen van afvlakking van de curves te zien. De hoogst behaalde speedup is 21.7 op 28 processoren, een respectabele efficiency van 78%. Als hierin voor het dual processor effect wordt gecorrigeerd dan is de efficiency zelfs iets groter dan 1.

Het opvallendste in de uitgesplitste efficiencies in Figuur 4 is de dip in de numerieke efficientie bij twee processoren. Deze blijkt werkelijk te bestaan. Bijvoorbeeld is de numerieke efficientie voor de drie runs op twee processoren van twee aparte nodes respectievelijk 0.937, 0.936 en 0.931. De drie runs zijn nogmaals herhaald, waarbij waardes 0.937, 0.929 en 0.930 zijn verkregen.

Voor het overige komt het gedrag in deze grafieken overeen met dat voor het CSM8 model. De numerieke efficientie is wat lager voor het model bak_parll, met name bij gebruik van 2 processors per node. De idle efficientie is juist beduidend beter, de berekeningen zijn een stuk beter gebalanceerd. Tenslotte is de communicatie efficientie hoger voor het bak_parll model, waardoor er ook een groter aantal processoren nuttig kan worden gebruikt.



Figuur 3: *Speedup voor het model "bak_parll" met TRIWAQ (*$360 \times 360 \times 4$ *punten), apart voor runs waarbij 1 processor per node en 2 processors per node worden benut.*



Figuur 4: *Uitsplitsing van de efficientie van de parallelle berekening voor het model bak_parll. Van boven naar beneden worden de relatieve rekensnelheid, de mate van load balancing, en de communicatie overhead getoond.*

5 Conclusies en aanbevelingen

In dit memo wordt verslag gedaan van de werkzaamheden van V$\mathcal{O}$Rtech Computing voor het realiseren van een MPI-versie van parallel WAQUA/TRIWAQ die geschikt is voor een Linux-cluster op basis van Myrinet.

De werkzaamheden zijn min of meer volgens plan uitgevoerd. Een praktische meevaller was dat er voor de ondersteuning van de MPI-versie MPICH-GM op het betreffende Linux-cluster slechts beperkte aanpassingen ten opzichte van de oorspronkelijke MPI-versie van parallel WAQUA/TRIWAQ hoefden te worden gemaakt. Deze aanpassingen aan de runprocedure `waqpro.run` kunnen in principe ook via een switch in de standaardversie worden geïntegreerd.

Met de nieuwe versie op basis van MPI wordt op het Linux-cluster van TUD-ct een uitstekende parallelle performance behaald. Voor het CSM8-model met TRIWAQ wordt op 8 processoren de uitzonderlijk hoge speedup van 6.9 behaald. Voor een ander testmodel met TRIWAQ dat speciaal voor performancetesten is opgezet wordt op 28 processoren een speedup van 21.7 bereikt. Gegeven het feit dat de gebruikte PC's last hebben van een behoorlijk groot dual processor effect (als er twee sequentiële sommen tegelijk op dezelfde node van het cluster worden gedraaid dan zitten ze elkaar in de weg en wordt slechts 76% van de maximale rekensnelheid behaald) is dit praktisch "ideale speedup".

In dit project zijn de MPI-versie en het nieuwe stopcriterium voor parallel WAQUA opnieuw geïmplementeerd en zijn de resultaten opnieuw met die van de standaardversie van WAQUA/TRIWAQ vergeleken voor de gehele testbank van SIMONA. Daarbij konden alle verschillen worden verklaard, waarmee is aangetoond dat de aanpassingen betrouwbaar functioneren. Het verdient aanbeveling dat ze ook in de moederversie van WAQUA/TRIWAQ in SIMONA worden geïntroduceerd.

Referenties

[1] E.A.H. Vollebregt. Testen met het Scalwest model op het Linux-cluster van WLH. Memo EV/M04.066, versie 1.0, V$\mathcal{O}$Rtech Computing, Juli 2004.

[2] E.A.H. Vollebregt. Verbetering van de performance van parallel WAQUA op het Linux rekencluster van RIZA-Arnhem. Technical Report TR04-03, V$\mathcal{O}$Rtech Computing, Postbus 260, 2600 AG Delft, Juni 2004.