Broich, Karl

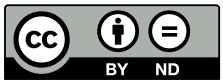
P_Flood - ein parallelisiertes Verfahren zur beschleunigten Berechnung von Wasserspiegellagen und Abflüssen

Dresdner Wasserbauliche Mitteilungen

Zur Verfügung gestellt in Kooperation mit/Provided in Cooperation with:
Technische Universität Dresden, Institut für Wasserbau und technische Hydromechanik

Verfügbar unter/Available at: https://hdl.handle.net/20.500.11970/103468

Vorgeschlagene Zitierweise/Suggested citation:
Broich, Karl (2014): P_Flood - ein parallelisiertes Verfahren zur beschleunigten Berechnung von Wasserspiegellagen und Abflüssen. In: Technische Universität Dresden, Institut für Wasserbau und technische Hydromechanik (Hg.): Simulationsverfahren und Modelle für Wasserbau und Wasserwirtschaft. Dresdner Wasserbauliche Mitteilungen 50. Dresden: Technische Universität Dresden, Institut für Wasserbau und technische Hydromechanik. S. 481-490.



Technische Universität Dresden – Fakultät Bauingenieurwesen
Institut für Wasserbau und Technische Hydromechanik

37. Dresdner Wasserbaukolloquium 2014
„Simulationsverfahren und Modelle für Wasserbau und Wasserwirtschaft“

P_Flood - ein parallelisiertes Verfahren zur beschleunigten Berechnung von Wasserspiegellagen und Abflüssen

Karl Broich

2d-hydraulische Berechnungen sind rechenintensiv. Die Grenzen der praktikablen Anwendung werden insbesondere bei Hochwassersimulationen oft erreicht. Die Konsequenzen sind zähe Arbeitsprozesse bei der Netzerstellung und lange Wartezeiten auf die Rechenergebnisse während der kritischen Projektendphase. Moderne PCs und Workstations verfügen heutzutage standardmäßig über mehrere Rechenkerne, die prinzipiell eine beschleunigte Berechnung ermöglichen. Wenn die Rechner in einem Netzwerk zusammengeschaltet sind, kann ein so genannter Cluster gebildet werden, der die vorhandene Rechenleistung nochmals erhöht. Um die verfügbare Leistung ausschöpfen zu können, ist jedoch eine anwendungsspezifische parallelisierte Programmierung notwendig. P_Flood nutzt hierzu das so genannte domain decompositioning. Hierbei wird das Berechnungsgebiet in Teilgebiete zerlegt. Diese Vorgehensweise erlaubt eine beschleunigte Berechnung und sorgt gleichzeitig durch kleinere Datenmengen innerhalb der Teilgebiete für ein effizientes Pre- und Postprocessing. Setup und Steuerung von P_Flood erfolgt über eine eigene Benutzeroberfläche. Das Berechnungsnetz wird mit dem bekannten SMS-Programm erzeugt. P_Flood ist auf Windows und Linux lauffähig.

Stichworte: 2d-hydraulische Berechnung, Parallelisierung Keywords 2d-SWE solver, parallelization, domain decompositioning, MPI, OpenMP

1 Rückblick

Initiiert wurde die technische Entwicklung von kostengünstigen Parallelrechnern durch den Bau des Beowulf-Cluster im Jahr 1995 (Becker et al.,1999). Hier wurde erstmalig der Nutzen der Standard-PC-Boards (commodity parts) für das Hochleistungsrechnen erkannt. Tatsächlich haben die Recheneinheiten von Standard-PCs eine Leistungsfähigkeit erreicht, die sich nicht wesentlich von speziellen Hochleistungsprozessoren von SGI oder SUN unterscheiden. In der Industrie werden PC-Cluster ebenfalls seit langem erfolgreich verwendet. Das beste Beispiel hierzu ist GOOGLE®.

Die Gesamtrechenleistung und Skalierbarkeit eines Clusters wird maßgeblich von der Prozessorleistung, der Netzwerkbandbreite und –durchsatz, von der Anzahl der Recheneinheiten und von der Parallelisierbarkeit der Anwendung bestimmt (Zitat Douglas Eadline, Clusterspezialist: „It all depends on your application.“).

Paralleles Rechen meint immer verteiltes Rechnen. Die Rechenaufgabe wird nicht von einer Rechenmaschine sondern verteilt auf mehrere Rechenmaschinen gleichzeitig gelöst. Ziel ist eine Reduzierung der Rechenzeit. Das bedeutet implizit, dass paralleles Rechnen nur dort Sinn macht, wo langwierige Berechnungen anfallen und die Zeit bis zur Ergebnisausgabe kritisch ist.

2d-hydraulische Berechnungen sind rechenintensiv, wenn sie ein großes Gebiet in hoher Auflösung oder lange Berechnungszeiträume umfassen. In den vergangenen 20 Jahren der Computerentwicklung galt das Mooresche Gesetz, wonach sich die Prozessorleistung alle 2 Jahre verdoppelt. Daher wurde mit jedem neuen Rechnerkauf ein ansehnlicher Rechenleistungsgewinn erzielt, der den Aufwand für eine Parallelisierung unwirtschaftlich machte. Dies gilt prozessorbezogen nicht mehr. Die Strukturen auf einem Chip sind derart klein geworden, dass sie fast die Wellenlänge des Lichts unterschreiten. Eine weitere Steigerung der Transistoranzahl pro Chip ist daher physikalisch nicht mehr möglich. Um dennoch eine Leistungssteigerung der Computer zu erzielen, werden daher Multicore Processor Units statt Central Processor Units verwendet. Schaltet man die Rechnereinheiten zusätzlich über ein Netzwerk zu einem Verbund zusammen, so erhält man einen so genannten Cluster, der die potentielle Rechenleistung nochmals steigert. Um die verfügbare Leistung ausschöpfen zu können ist eine parallelisierte Programmierung notwendig.

2 Beschreibung des Verfahrens

2.1 Numerisches Lösungsverfahren

Grundlage der Berechnung bilden die vollständigen Saint-Venant-Gleichungen. Diese werden auch Flachwassergleichungen genannt. Es handelt sich hierbei um partielle Differentialgleichung von hyperbolischem Typ. Hyperbolische Differentialgleichungen können in der Lösung Sprungstellen enthalten. Im wasserwirtschaftlichen Praxis erhalten diese bei der Ausbreitung von Schwall- und Sunkwellen sowie bei jedem Fließwechsel Bedeutung (Wechselsprung). Eine weitere Problematik der Modellierung besteht in der Benetzung und dem Trockenfallen von Flächen. Dieser Vorgang ist in natürlichen Flüssen in Folge der stetig wechselnden Wasserführung stets zu beobachten. Die ohnehin aufwändige

numerische Simulation wird durch die hieraus resultierende Notwendigkeit das Berechnungsgebiet (Nasszellen) ständig zu prüfen und anzupassen, verkompliziert. Daher können die erforderlichen Rechenzeiten u.U. unwirtschaftlich lang werden. Die nachfolgend beschriebene Parallelisierung der Numerik schafft hier Abhilfe.

Zur Dämpfung etwaiger Oszillationen wird oft künstliche Viskosität verwendet. Die Einarbeitung von Turbulenzmodellierung hat ebenfalls stabilisierende Wirkung. Aus theoretischen Gründen (Turbulenz und Rauheit lassen sich in der 2d-Simulation nicht trennen) und praktischen Erwägungen (Rechenzeitgewinn bei ausreichender Genauigkeit und Stabilität) wird auf beides verzichtet.

Das Programm P_Flood löst die Gleichungen mit einem expliziten Finite-Volumen-Verfahren auf Basis eines unstrukturierten Netzes aus Dreiecks- und Viereckselementen (Nujic, 1995). Explizite Gleichungslöser konvergieren langsamer gegen einen stationären Zustand, sind jedoch bei hochinstationären Verhältnissen - wie zum Beispiel bei Hochwasserwellen und Überflutungen - deutlich genauer. Die Gleichungen werden in zwei Schritten nach dem McCormack-Verfahren gelöst (Prediktor-Korrektor-Verfahren). Durch Einteilung des Gebiets in Untergebiete wird eine Verteilung der Rechenlast auf mehrere Rechenkerne möglich. Diese Form der Parallelisierung wird als domain decompositioning bezeichnet. Sie bewirkt auf modernen Multi-Core-Computern oder Verbundrechnern (Cluster) eine Reduzierung der Rechenzeit.

Die Flachwassergleichungen lassen sich aus den allgemeinen Strömungsgleichungen nach Navier-Stokes durch Tiefenintegration unter Annahme hydrostatischer Druckverteilung und konstanter Geschwindigkeit herleiten. Sie setzen sich aus den Bilanzgleichungen für Volumenkontinuität und Impuls zusammen. Beide Gleichungen sind gekoppelt. Die Impulsgleichung ist vektoriell und muss daher für x- und y-Richtung separat angesetzt werden. Für die 2d-Berechnung sind daher 3 gekoppelte Gleichungen zu lösen. Das Gleichungssystem ist durch Quellterme für Sohlneigung, Reibung sowie lokale Zu- und Abflüsse zu ergänzen. Die Gleichungen werden in konservativer Form geschrieben und gelöst.

Der Quellterm aus Neigung $I_{So} = gh \cos\alpha \sin\alpha$ wird üblicher Weise unter der Annahme $\cos\alpha \cong 1$, $\sin\alpha \cong \tan\alpha = I_{So}$ linearisiert. Hierdurch können jedoch im Bereich von Böschungen und Abstürzen Diskretisierungsfehler entstehen. Der Neigungsterm wird daher mit einer höheren Genauigkeit berücksichtigt.

Zur Berechnung des Reibungsgefälles I_R wird eine Fließformel benötigt. Die Fließformel nach Manning/Strickler findet breiteste Anwendung in der Praxis, obwohl bekannt ist, dass sie fließtiefenabhängig ist. Deren Rauheitswert k_{st} ist

für verschiedene Sohlenmaterialien, Sohlenformen und Fließverhältnisse gut dokumentiert.

Die räumliche Diskretisierung geschieht mittels Finiten Volumen. Hierbei muss unterschieden werden zwischen knoten- und zell-zentrierter Formulierung. Bei der zell-zentrierter Formulierung liegen die rechnerischen Kanten auf der Elementberandung und der Rechenknoten im Zellschwerpunkt. Für die knotenzentrierte Formulierung ist eine Delaunay-Triangulation um den Knoten herum erforderlich. P_Flood verwendet die knoten-zentrierte Formulierung. Dies hat den Vorteil, dass die Eingabe- und Ausgabeknoten den tatsächlichen Berechnungsknoten entsprechen. Es ist zu beachten, dass die Elementkanten nicht den rechnerischen Kanten entsprechen.

Das Verfahren ist 1. Ordnung Genauigkeit bezüglich des Orts. Die Strömungszustandsvektoren sind daher rechnerisch innerhalb jeder Zelle konstant. Dies führt zu Sprungstellen auf den rechnerischen Kanten. Der sogenannte Riemann-Gleichungslöser sorgt hier für einen Ausgleich. Es gibt verschiedene Verfahren zur Lösung des Riemann-Problems. P_Flood verwendet den Ansatz nach Lax-Friedrich.

Die numerische Lösung bezüglich des Orts ist nun bekannt. Für die Berechnungsfortführung in Zeitrichtung wird das Zeitschritt-Verfahren nach MacCormack verwendet. Es handelt sich hierbei um ein Prediktor-Korrektor-Schema mit 2. Ordnung Genauigkeit.

Weitere Details zum Verfahren können im P_Flood-Handbuch nachgelesen werden (Bezugsquelle: www.ib-broich.de/kundeninformation).

Die prinzipielle Richtigkeit des Verfahrens wurde anhand zahlreicher praxisrelevanter Beispiele nachgewiesen (Bezugsquelle: www.ib-broich.de/kundeninformation).

2.2 Parallelisierung

Die Nutzung der Leistungsvorteile der Multi-Core-Rechner und der Cluster erfordert spezielle Programmierung. Programme ohne Parallelisierung laufen auf modernen Rechnern nicht deutlich schneller als auf alten, weil sie nur einen Rechenkern nutzen können.

Die Parallelisierung eines Programms kann auf verschiedenen Ebenen ausgeführt werden:

- Die einfachste Form der Parallelisierung besteht im wiederholten Aufruf eines Programms bei unterschiedlichen Daten (embarrassingly parallel). Dies geschieht in Form eines Batch-Betriebs.

- Teile des Quellcodes lassen sich automatisiert parallelisieren mittels IPO (interprocess optimization). Diese Art der Optimierung kann durch eine Compileroption aktiviert werden.
- Eine effizientere Parallelisierung kann durch spezielle OpenMP-Programmierung erzielt werden. OpenMP (Open Multi-Processing) teilt im wesentlichen for-Schleifen in so genannte Threads (verteilbare kleine Rechenaufgaben) auf. OpenMP-Prozesse nutzen Multi-Core-Rechner mit gemeinsamen Hauptspeicher (=Shared Memory Processor SMP) , Cluster werden aber nicht unterstützt.
- Die obere Rechenebene, sprich der Algorithmus, kann vom Compiler nicht in seiner Struktur erkannt und zerlegt werden. Hier ist eine aufwändige Analyse und Umarbeitung der Programmstruktur erforderlich. Das Programm wird entsprechend seines Aufbaus spezifisch parallelisiert. Für P_Flood wurde das sogenannte domain-decompositioning mittels MPI-Standard-Schnittstelle verwendet. Hierbei wird das Berechnungsnetz in gleichgroße Teilgebiete (Kacheln) zerlegt und ein Datenaustausch über die gemeinsamen Ränder hinweg mittels MPI (Message Passing Interface) implementiert. MPI-Programme nutzen Multi-Core-Rechner und Cluster.
- Falls der Computer über Cuda-fähige Grafikkarten oder Komponenten verfügt, kann eine Rechenbeschleunigung mittels Cuda-Programmierung erzielt werden. Die Cuda-Programmierung funktioniert nicht clusterübergreifend.

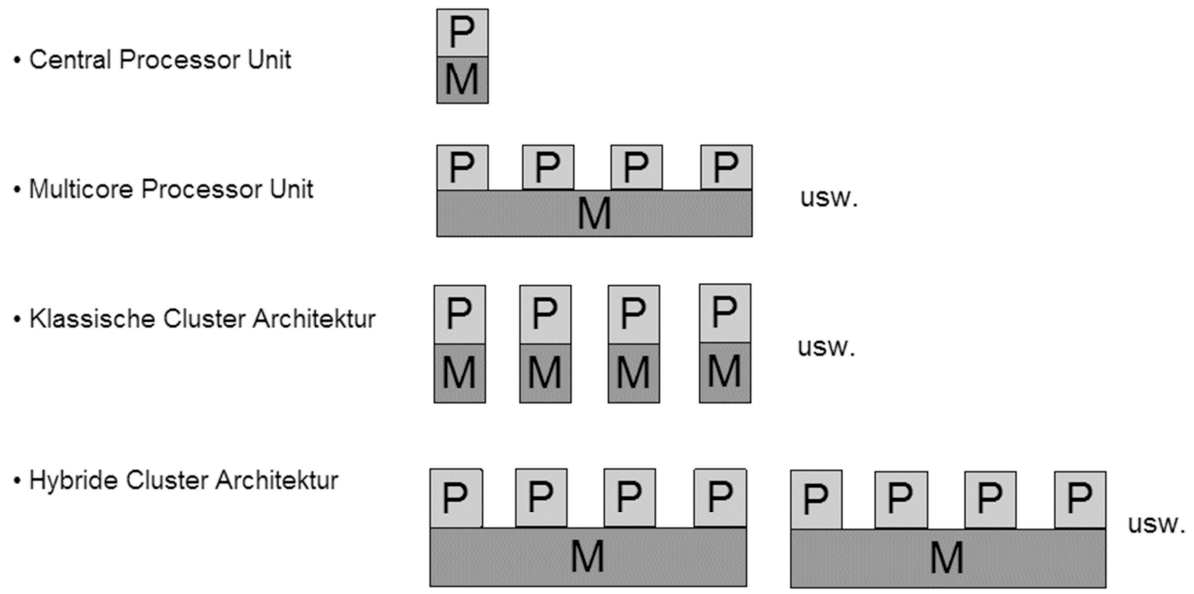


Abbildung 1: Nutzung der Clustertechnologie

P_Flood kann für die erstgenannten vier Arten der Parallelisierung betrieben werden. Cuda-Programmierung wird derzeit nicht verwendet, weil dies spezielle Hardwarekomponenten voraussetzt. Die einfache serielle 2d-hydraulische Berechnung ist mit P_Flood weiterhin möglich.

2.3 Implementation und Design

Programmiersprachen

Für die Programmierung der *Kernmodule* (P_Flood_Engine) wurde die Programmiersprache FORTRAN90 beibehalten. Sie hat gegenüber der C++-Programmierung den Vorteil bis zu 30 Prozent schnelleren Code zu erzeugen. P_Flood_Engine besteht aus den folgenden Modulen:

- P_Flood_Core : Gleichungslöser,
- P_Flood_Prepro : Datenaufbereitung, serielle Berechnung,
- P_Flood_Prepar : Datenaufbereitung, parallele Berechnung,
- P_Flood_Premet : Datenteilung,
- P_Flood_Prestr : Detailnetzbildung für 3d-Ansicht.

Für die *Benutzeroberfläche* (P_Flood_Control, P_Flood_Setup, P_Flood_Constructor) wurde eine Kombination aus Python- und Qt-Programmierung verwendet. Python ist eine objektorientierte Skriptsprache, die im wissenschaftlichen Umfeld und in der Geoinformatik weitverbreitet ist. Die Grafikbibliothek Qt ist Grundlage von KDE (Linux-Windowsoberfläche) und somit ebenfalls sehr verbreitet.

Für die Ausführung der *Stapelbefehle* wird je nach Betriebssystem entweder die DOS- oder Bash-Shell verwendet.

Generell sind die Programme so gestaltet, dass sie mit mäßigen Veränderungen sowohl auf Windows als auch auf Linux lauffähig sind.

Pre- und Postprocessing

Die Vorbereitung der Berechnung geschieht im Wesentlichen mit dem Pre- und Postprocessor SMS (Bezugsquelle: www.aquaveo.com) und P_Flood_Setup. In der Praxis sind zur Steigerung von Produktivität und Qualität zusätzliche Programme zur Datenausdünnung und Konstruktion von Flussschlauch, Dämmen. Wehren und Durchlässen notwendig.

Parallelisierung

Die Methode des domain decompositioning ist eine von mehreren möglichen Arten der Parallelisierung. Sie wurde aus praktischen Gründen gewählt, weil sie

sowohl die Ausführung des Programms als auch den Prozess der Datenaufbereitung beschleunigt und somit doppelt wirkt. Ab etwa 1.000.000 Knoten wird der Bildaufbau mit dem Pre- und Postprocessor SMS träge und die Bearbeitungsgeschwindigkeit sinkt. In diesem Fall lohnt dann eine Netzteilung allein schon aus Gründen der Datenverarbeitung. Die Größe der Teilgebiete liegt in der Regel bei 100.000 bis 500.000 Knoten. Das domain decompositioning wurde mit Hilfe von MPI- Programmierung umgesetzt. Einzelne häufig durchlaufene For-Schleifen sind zusätzlich mit OpenMP parallelisiert.

3 Benchmark-Ergebnisse

Mit dem Programm P_Flood wurden Testberechnungen für verschiedene Programmvarianten durchgeführt. Hierbei wurden Berechnungsbandbreite, Compileroptionen und Art der Parallelisierung variiert. Die Programmvarianten wurden mit dem INTEL®-Fortran-Compiler auf Intel-Xeon-Dual-Quad-Prozessoren erzeugt. Die u.g. Resultate gelten für diese Plattform. Die Optimierung von parallelen Programmen ist aber hochgradig hardwarespezifisch. Für andere Rechner und Compiler sind daher die Benchmarks zu wiederholen. Der Rechenzeitunterschied für ein gut und schlecht optimiertes Programm kann durchaus 100 Prozent und mehr betragen.

3.1 Annahmen und Randbedingungen

Die Bandbreite ist bei numerischen Berechnungen bedeutend. Der Umstieg von 32bit auf 64bit Bandbreite bringt eine Rechenbeschleunigung von etwa 20 Prozent für die vorliegende Anwendung. P_Flood wird daher standardmäßig für 64bit-Bandbreite kompiliert.

3.2 Compileroptionen

Im Vergleich zur Standardkompilierung ohne Compileroptionen ergab die Kompilierung mit der Option -O3 und IPO-Compileroption nur eine geringfügig schnellere Programmausführung.

3.3 OpenMP

Der INTEL®-Compiler erkennt OpenMP.-in-line-Direktiven. Die Parallelisierung wird im Rechenregister ausgeführt. Im Vergleich zur Standardkompilierung ergibt sich eine um 10% schnellere Programmausführung je Prozessor. Dies entspricht einem Parallelisierungsgrad von P_{SMP} =0,72.

3.4 MPI

Die Verwendung von MPI erfordert immer einen Eingriff in den Berechnungscode. Im vorliegenden Fall war die Zerlegung des Berechnungsgebiets das Ziel und das Programm musste hierzu in weiten Teilen neu geschrieben und ergänzt werden, um den Datenaustausch zwischen den Prozessen entlang der gemeinsamen Berandung der Teilgebiete so zu gestalten, dass die Numerik dort unbeeinflusst bleibt. Bei sinnvoller Teilung des Berechnungsgebiets kann ein Parallelisierunggrad von $P_{Cluster} = 0{,}76$ bei Überflutungsberechnungen erzielt werden. Bei stationären Berechnungen kann eine höhere Effektivität erzielt werden.

3.5 Hybrid OpenMP/MPI

OpenMP und MPI sind nicht orthogonal. D.h. sie beeinflussen sich gegenseitig. Die erzielbare Rechenbeschleunigung ist daher abhängig von der Verteilung der Teilgebiete auf die Rechner des Clusters.

3.6 Erzielbare Rechenbeschleunigung

Die erzielbare Rechenbeschleunigung (Speedup) steigt nicht linear mit der Prozessoranzahl, sondern ist nach dem Amdahl'schen Gesetz (1967) begrenzt. Sie ist demgemäß in Abhängigkeit von Parallelisierungsgrad P und Prozessoranzahl N wie folgt definiert:

$$s = 1 \,/\, \left(\left(1 - P\right) + P / N \right) \quad . \tag{1}$$

Die theoretisch maximal erzielbare Rechenbeschleunigung ist 1/(1-P). Sie ist unabhängig von der Prozessoranzahl.

Für einen Cluster mit 32 Berechnungsknoten zu jeweils einem Prozessor beträgt im Vergleich zum seriellen 64bit-Programm: die Rechenbeschleunigung

$$s_{Cluster} = 1 \,/\, \left(\left(1 - P_{Cluster}\right) + P_{Cluster} / N_{Cluster} \right) = 3{,}79 \tag{2}$$

Für einen hybriden Cluster mit 32 Berechnungsknoten zu jeweils 8 Prozessoren gilt:

$$\begin{aligned} s_{hybrid} = s_{Cluster} \cdot s_{SMP} &= 1 / \left(\left(1 - P_{Cluster}\right) + P_{Cluster} / N_{Cluster} \right) \\ &\quad \cdot 1 / \left(\left(1 - P_{SMP}\right) + P_{SMP} / N_{SMP} \right) \\ &= 3{,}79 \cdot 1 / \left(\left(1 - 0{,}72\right) + 0{,}72 / 8 \right) \\ &= 3{,}79 \cdot 2{,}70 = 10{,}72 \end{aligned} \quad . \tag{3}$$

Wenn die Anzahl der Berechnungsknoten verdoppelt wird beträgt der Speedup 3,97 2,70 = 10,72. Der Unterschied beträgt nur 4,7%. Aus diesem Zusammen-

hang folgt, dass die Wirtschaftlichkeit eines Clusters mit steigender Knotenanzahl stark sinkt. Die wirtschaftliche Knotenanzahl sollte daher in der Planungsphase festgelegt werden.

Die erzielbare maximale Rechenbeschleunigung liegt demnach relativ konstant bei etwa 10. Hieran wird sich mittelfristig nicht viel ändern.

Die Aufteilung in Teilgebiete hat den zusätzlichen Nutzen, dass deutlich größerer Gesamtgebiete simuliert werden können. Der genannte hybride Cluster wäre zum Beispiel in der Lage ein Flusssystem mit etwa 1000 km Fließstrecke hochgenau (20.000 Netzknoten/km) zu berechnen. Hierdurch wird die 2d-hydraulische Simulation in der Fläche auf den Skalenbereich einer NA-Simulation erweitert.

4 Zusammenfassung

In der praktischen Anwendung gerät die 2d-hydraulischer Simulation an ihre Grenzen. Die Ursachen hierfür liegen oft nicht in mangelnder Leistungsfähigkeit der verwendeten Rechner, sondern in unzureichender Auslastung der Rechner durch die verwendeten Simulationsprogramme. Das 2d-Simulationsprogramm P_Flood sorgt durch Parallelisierung mit domain decompositioning für eine bessere Auslastung vorhandener Computerhardware, Dies bewirkt eine beschleunigte Berechnung und ermöglicht deutlich größere Berechnungsgebiete. P_Flood bietet eine effiziente Lösung für komplexe Hydraulik, eröffnet aber auch neue Möglichkeiten für flächendeckende hochgenaue Berechnung, wie z.B. die großräumige 2d-Hochwassersimulation oder die Simulation von Starkregenereignissen in höchster Detailauflösung.

5 Literatur

Amdahl, Gene M. (1967): Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities, *AFIPS Conference Proceedings* (30): 483–485. doi:10.1145/1465482.1465560

Donald J. Becker, John Salmon, Daniel F. Savarese and Thomas Sterling (1999): How to Build a Beowulf - A Guide to the Implementation and Application of PC Clusters, MIT Press, 1999

Nujic, M., (1995): Efficient Implementation of Non-oscillatory Schemes for the Computation of Free Surface Flows, Journal of Hydraulic Research, 33(1),1995

Autor:

Dr.-Ing. Karl Broich

Ingenieurbüro Broich
Torriweg 45
81247 München

Tel.: +49 89 420950904
Fax: +49 32223712979
E-Mail: karl.broich@ib-broich.de