Power Grid Exist:

A Dynamic Cluster with Hierarchic Master Worker Architecture

Prof. Dipl.-Inform. Günther Bengel

Informatik-Berichte
Hochschule Mannheim – Fakultät für Informatik

Computer Science Reports
Mannheim University of Applied Sciences – Computer Science Department

CSR 001.08
August 2008

URL: http://www.informatik.hs-mannheim.de/reports

hochschule mannheim

FAKULTÄT FÜR INFORMATIK

Prof. Dipl.-Inform. Günther Bengel

Hochschule Mannheim
Institut für Betriebssysteme
Fakultät Informatik

Paul-Wittsack-Str. 10
D-68163 Mannheim

Tel.: +49 (621) 292 6223
Fax.: +49 (621) 292 6237

http://www.bts.hs-mannheim.de
http://www.vts.hs-mannheim.de
http://www.pvs.hs-mannheim.de

email: g.bengel@hs-mannheim.de

1. Einführung Power Grid Exist

Power Grid Exist (PGE) ist ein Cluster aus unterschiedlichen größtenteils bestehenden, vorhandenen und sich in Betrieb befindlichen Rechnern (PC, Workstations, Server, Cluster), das die brachliegenden und nicht genutzte Rechenleistung ausnutzt. Dementsprechend entspricht PGE einem Feierabend-Cluster [BBKS 08]. Feierabend-Cluster sind schon seit Jahren bei der BASF Ludwigshafen im Einsatz und nutzen die bei Unternehmen, wie an der Hochschule auch, die vorhandenen brachliegenden Rechnerressourcen aus. Bei diesem Vorgehen sind die TCO-Kosten am geringsten, da ja die Rechner alle laufen und die Wartung von den einzelnen Instituten, von der Hochschule und Forschungszentren und den externen Anbieter übernommen werden.

PGE ist dynamisch: Je nach vorliegender Problemgröße und Arbeitslast können weitere Rechner hinzugeschaltet werden, bis von der Laufzeit her eine optimale Problemgröße und Arbeitslast vorliegt. PGE ist ein *atmendes Cluster*, das je nach Anforderung und Bedarf Rechnerressourcen hinzu und wieder abschaltet.

Der Aufbau von PGE geschieht in mehreren Stufen und Schritten:

1. Stufe: Für die Grundlast und zur Softwareentwicklung für PGE steht eine Clustermaschine zur Verfügung.

2. Stufe: Alle Rechner des Instituts für Betriebssysteme.

3. Stufe: Alle Rechner der Fakultät Informatik.

4. Stufe: Weitere Rechner der Hochschule Mannheim.

5. Stufe: Einbindung von externen Rechnern (Forschungszentrum Karlsruhe, Universität Heidelberg, Berufsakademie Mannheim, externe Anbieter).

Jeder Rechner in PGE kann als Zugang zu einem Eingangsportal von PGE genutzt werden und ist selbst auch wieder ein Bestandteil von PGE.

2. Architektur von PGE

2.1 Ausgangspunkte

Die meisten Grid-Applikationen liegen heute noch nicht als Grid-Service vor und verbergen nicht das dahinterliegende Grid und Cluster. Ein Anwender muss

- die Parallelisierung des Problems selbst vornehmen, und er braucht somit Kenntnis von
- der Grid-Middleware zur Parallelisierung und Lösung seines Problems und
- damit Kenntnisse vom Aufbau des Grid und somit des Clusters.

Zur Vermeidung dieser Kenntnis ist PGE nach einem gebräuchlichen Parallelisierungsschema dem *Master Worker-Schema* aufgebaut. Mit diesem Schema lassen sich die meisten Probleme parallelisieren und verschiedene mathematische Modellierungen und Algorithmen aus verschiednen Bereichen werden daraufhin evaluiert und auf dieses Schema übertragen (siehe Abschnitt 3). Dem Master Worker–Schema zur Lastverteilung folgt auch das aus dem Internet bekannte SETI@home-Projekt (Search for extraterrestrial intelligence at home, englisch für „Suche nach außerirdischer Intelligenz zu Hause“) [ACK 02].

Ein einziger Eingang und somit ein Eingangsportal in PGE birgt folgende Vorteile:

- Der Benutzer braucht sich nicht um die Parallelisierung des Problems zu kümmern, sondern die parallele Lösung liegt mit dem implementierten Master Worker-Schema vor.
- Die Parallelisierung des Problems ist vorgegeben und der Benutzer braucht keine Kenntnisse von der Grid Middleware.
- Mit einem einzigen Eingang zu dem Schema lassen sich die Problemlösung leicht als Grid-Service und somit Web-Service formulieren. Damit kann der Service weltweit über das Internet genutzt werden. Dass zur Abwicklung des Service ein Grid, und somit ein Cluster benutzt wird, ist vor dem Benutzer verborgen und er sieht nur den Portalrechner dem er das Problem und seine Daten übergibt und von ihm wieder die Ergebnisse entgegen nimmt.

2.2 Master Worker-Schema

Bei Master Worker verteilt ein *Master* die verschiedenen Datenbereiche an eine bestimmte Anzahl von *Workers* und nimmt die Ergebnisse von den Workers entgegen (siehe dazu auch [BBKS 08]).
Nach Programmstart fragen alle Workers beim Master nach einem unerledigten Datenbereich an. Der Master nimmt von einem Stapel die unerledigten Datenbereiche und weist sie den einzelnen Worker zu. Nachdem der Worker den aktuellen Datenbereich bearbeitet hat, schickt er das Ergebnis an den Master zurück und fragt nach einem nächsten zu bearbeitenden Datenbereich. Der Overhead der Parallelisierung ist hier die Rüstzeit, also die Zeit zum Start des Programms und das Verteilen der Datenbereiche und das Einsammeln der Ergebnisse.

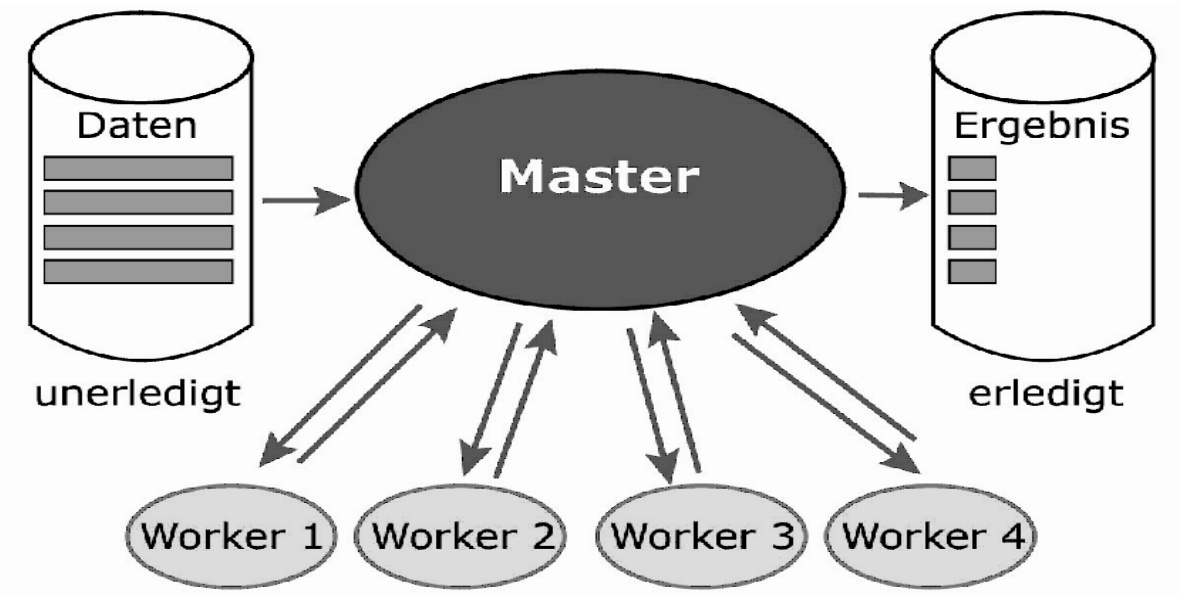

Abb. 2-1: Master Worker-Schema

In einem ersten Ansatz setzen wir eine Regularität der Daten und Uniformität der Arbeit voraus, d.h. für alle Teildaten wird die gleiche Anzahl von Rechenoperationen benötigt, und alle Algorithmen sind gleich und laufen auf gleich schnellen CPUs. Vernachlässigt man die Kommunikation zwischen dem Master und den Workers und die Arbeit des Workers, so ist mit folgendem Speedup zu rechnen:

Dabei sei n sei die Größe der einzelnen Datenzerlegungen und p die Anzahl der Workers

$$T'(n) = p * n$$

Die Laufzeit im parallelen Fall, also die Laufzeit eines einzelnen Workers ist:

$T_p(n) = n$

Somit erbringt das Master Worker-Schema folgende Leistungssteigerung:

$$S_p(n) = \frac{T'(n)}{T_{p(}n)}$$

$$= \frac{p * n}{n}$$

$$= p$$

Bei den Speedup-Betrachtungen wurde die Kommunikation zwischen dem Master und den Worker außer Acht gelassen. Je feiner man die Granularität der Datenzerlegung wählt, umso mehr steigt die Kommunikation zwischen dem Master und den Workers an, die zu übertragenden Daten nehmen jedoch ab und die Arbeitslast des Masters steigt an. Dies kann bei sehr feiner Granularität der Daten sogar auf eine Überlastung des Masters führen und der Master wird zu einem leistungsbeschränkenden Flaschenhals.

Mit dem Master Worker-Schema lassen sich viele Grid-Applikationen aus verschiedenen Bereichen, die heute noch mit Grid-Middleware gelöst werden auf schnelle und kostengünstige Weise in Grid-Services überführen.

2.3 PGE Aufbau und Architektur

2.3.1 Cell Broadband Engine Architecture (CBEA)

Der Grundbaustein von PGE orientiert sich an der Cell Broadband Engine Architecture (CBEA). Toshiba, Sony und IBM entwickeln seit dem Jahr 2000 gemeinsam einen Prozessor namens *Cell*, der in der Playstation 3 läuft, und in HDTV-Geräten und Servern eingesetzt werden soll [C 07] [KDH 05]. Die *Cell Broadband Engine Architecture* (*CBEA*) besitzt ein 64 Bit PowerPC Kern (Power Processor Element (PPE)) und acht speziell ausgelegten „Synergistic“ Kernen (Synergistic Processor Elements (SPE)) auf einem Chip, die mit einem Hochgeschwindigkeits-Bus (Element Interconnect Bus (EIB)) verbunden sind. Zusätzlich auf dem Chip integriert ist ein Hochgeschwindigkeits-Speicher- und ein -I/O-Interface.

Das Power Processor Element (PPE) besitzt eine 32-Kbyte Instruktions- und Datencache und einen 512-Kbyte großen einheitlichen Cache auf der zweiten Ebene. Zur Reduktion der Hauptspeicherzugriffe der acht *Synergistic Processor Elements* (*SPE*) besitzen sie keinen Cache, sondern einen 256-Kbyte großen lokalen Speicher. Zum Datentransfer zwischen dem lokalen Speicher und dem gemeinsamen Hauptspeicher über den Element Interconnect Bus (EIB) [KPP 06] besitzt jede SPE einen Memory Flow Controller (MFC).

Synergistische Architekturen [GHF 06] sind Datenparallelität ausnutzende Architekturen und unterstützen einen hohen Thread Level-Parallelismus durch eine Vielzahl von Prozessoren auf einem Chip.

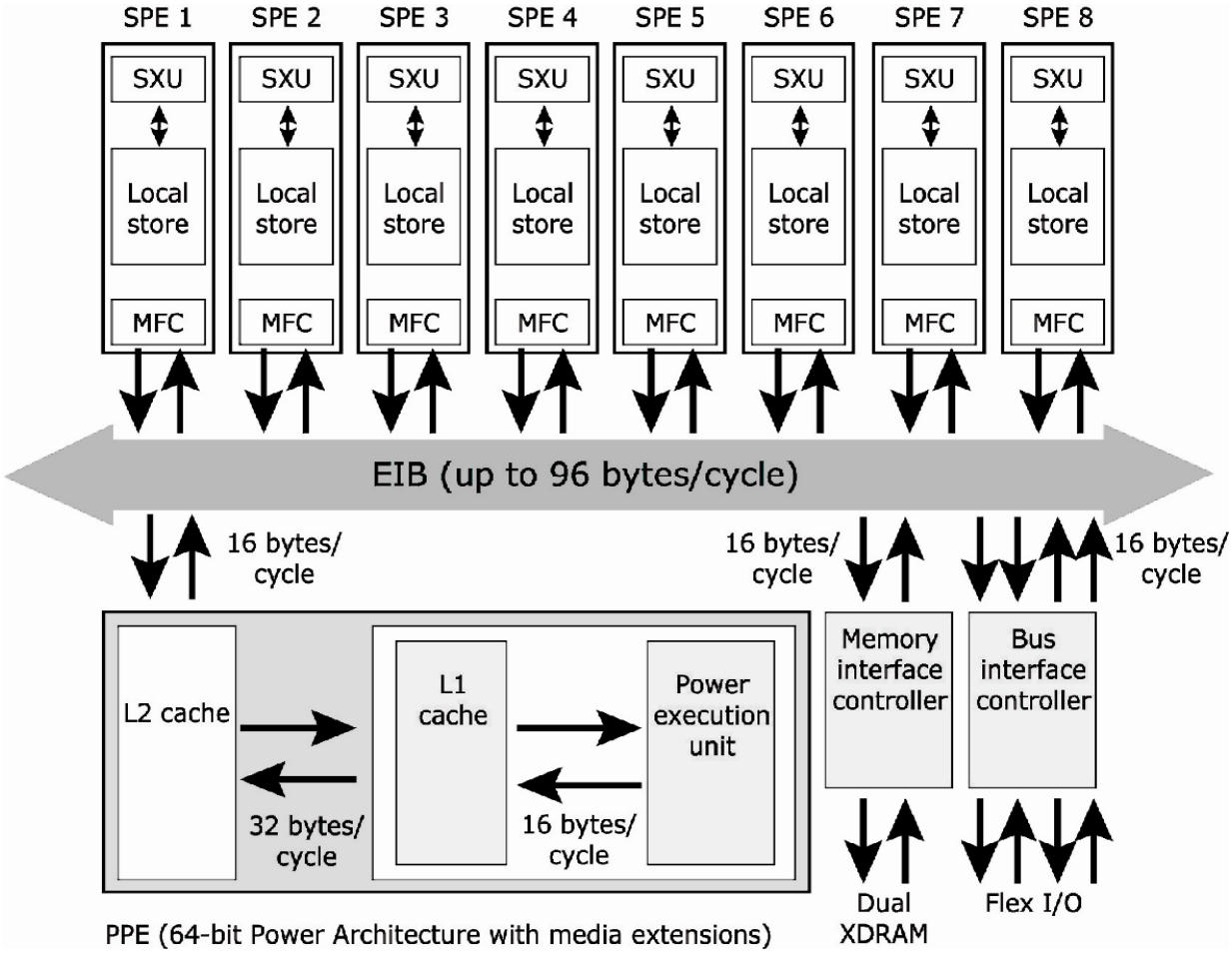

Abb. 2.2: Cell Broadband Engine Architecture (CBEA)

Das Betriebssystem für die Cell Broadband Engine Architecture ist *Cell Linux*. Cell Linux ist, wie die restliche Cell-Software, Open Source. Der gegenwärtige Stand von Cell Linux für die heterogenen Prozessoren, des Laders, der Compiler und Compilerprototypen und der Tools und des Debuggers ist in [GEM 07] beschrieben.

Grundelement des Aufbaus von PGE ist also eine Architektur bestehend aus 8 Workers und 1 Master (1 M + 8 W), also neun Rechnern, und dies ist ein Cluster erster Stufe.

2.3.2 Hierarchische Master-Worker Architektur von PGE

Die acht Workers und ein Master bilden den Grundbaustein von PGE. Damit lassen sich dann schrittweise weitere Rechner mit in das Cluster PGE aufnehmen. Durch Hinzufügen eines weiteren Masters (MasterMaster) an der Spitze, kann dann dieser MasterMaster wieder bis zu 8 weiteren Masters mit jeweils 8 Workers bedienen. Dies Cluster ist ein Cluster zweiter Stufe und besteht im Maximalausbau aus 1 MasterMaster und bis zu 8* 1 M + 8 W also aus 1 MM + 8 * (1 M + 8 W) = 73 Rechner.

Mit einem MasterMasterMaster ergibt sich dann ein Cluster der Stufe drei und dieses Cluster besitzt im Maximalausbau 1 MMM + 8 * (1 MM + 8 * (1 M + 8 W)) = 585 Rechner.

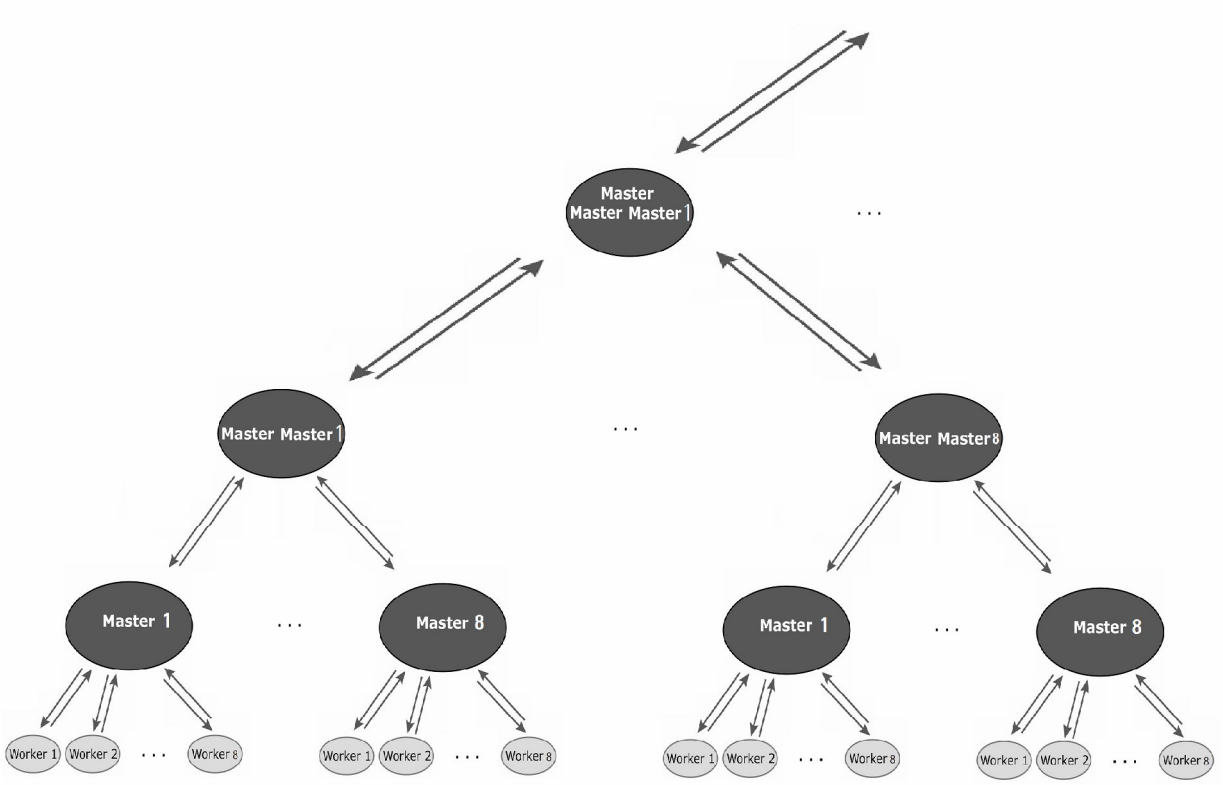

Abb. 2-3: Cluster der Stufe drei mit hierarchischer Master Worker-Architektur

Wie viel Rechner für ein Cluster der Stufe vier, fünf, sechs, … im Maximalausbau benötigt werden, kann der Leser selbst ermitteln und die Anzahl der Rechner ist nach oben nicht beschränkt.

Bei diesem Aufbau des Clusters ist jedoch zu beachten, dass nicht die Master…Master zu einem leistungsbeschränkenden Flaschenhals werden. Des Weiteren bilden die Master…Master ein einzelner Ausfallpunkt in dem Cluster und sind dementsprechend redundant und fehlertolerant auszulegen.

3. Anwendungen

Zum Nachweis der Anwendbarkeit und zur Feststellung der Güte des hierarchischen Master-Worker-Konzeptes, des Laufzeitverhaltens der Applikationen und Messung des gewonnenen Speedup dienen die folgenden Anwendungsgebiete:

- Finiten Element-Methode (FEM),
- Monte Carlo Simulation und Risikoanalyse,
- Gebiete der Kryptographie,
- Gebiet des Suchens, Suchalgorithmen, Suchmaschinen, Invertierte Index Suche,
- Gebiet des Sortierens, Sortieralgorithmen,
- ….

3.1 Hierarchische Master Worker-Schema -> Grid-Service

Die Überführung der obigen beschriebenen Grid-Applikationen in Grid-Services geschieht in zwei Schritten:

1. Überführung des Master Worker-Schemas in einen Grid-Service.

2. Anpassung der verschiedenen Probleme aus oben beschriebenen Gebieten an das Master Worker-Schema und Gewinnung von speziellen Grid-Services.

3.2 Master Worker-Schema -> Grid-Service

Die Realisierung geschieht in folgenden Schritten:

1. Implementierung des Master Worker-Schemas in C (C-Bibliotheken liegen dazu vor) [BM 06].
2. Laufzeitmessungen am Master Worker Schema mit unterschiedlichen Problemgrößen und unterschiedlichen Lasten für Master und Worker zur Bestimmung der optimalen Parallelitätsgranularität und der optimalen Anzahl der Workers.
3. Aus den Laufzeitmessungen Erstellung der Service-Levels.
4. Optimale Parallelitätsgranularität erreicht und alle Service-Levels liegen fest? Nein, gehe zu Schritt 2.
5. Implementierung des Worker-Schemas als Web-Service, wobei der Anwender den Master- und Worker-Code herunterladen muss.

3.3 Finiten Element Methoden -> Spezielle Grid-Service

Die Realisierung geschieht in folgenden Schritten:

1. Anpassung der finiten Element Methoden an das Master Worker Schema.
2. Entwickeln des Master und Worker Codes.
3. Einhängen des Master Worker Codes in das in Schritt 1 entwickelte Master Worker-Schema.
4. Erstellen und Anpassen des Web-Service (Schritt 1) an die speziellen Gegebenheiten.

3.4 Kryptographie-Algorithmen -> Spezielle Grid-Service

Die Realisierung geschieht in folgenden Schritten:

1. Anpassung der Kryptographie-Algorithmen an das Master Worker Schema.
2. Entwickeln des Master und Worker Codes.
3. Einhängen des Master Worker Codes in das in Schritt 1 entwickelte Master Worker-Schema.
4. Erstellen und Anpassen des Web-Service an die speziellen Gegebenheiten

Literatur

[ACK 02] Anderson D. P., Cobb J., Horpela E., Lebofsky M., Werthimer D.: Seti@home: An Experiment in Public-Resource Computing. Communications of the ACM, Vol. 45, No. 11, Nov. 2002.

[BBKS 08] Bengel G., Baun C. , Kunze M., Stucky K-U.: Masterkurs Parallele und Verteilte Systeme. Vieweg+Teubner Verlag 2008.

[BM 06] Bauke H., Mertens S.: Cluster Computing. Springer Verlag, 2006

[C 07] The Cell Chip: Informationen über den Multi-Core-Prozessor. http://www.the-cell-chip.de, 2007.

[GEM 07] Gschwind M., Erb D., Manning S., Nutter M.; An Open Source Environment for Cell Broadband Engine System Software. IEEE Computer, Vol. 40, No. 6, 2007.

[GHF 06] Gschwind M., Hofstee H.P., Flachs B., et. al.: Synergistic Processing in Cell's Multicore Architecture. IEEE Micro, Vol. 26, No. 2, March/April 2006.

[KDH 05] Kahle J. A., Day M. N., Hofstee H.P. et. al.: Introduction to the Cell Multiprocessors. IBM J. Research and Development, Vol. 49, No. 4/5, 2005.

[KPP 06] Kistler M., Perrone M., Petrini F.: Cell Multiprocessor Communication Network: Built for Speed. IEEE Micro Vol. 26, No. 3, May/June 2006.