Projektgruppenantrag

1 Thema

CyPhyControl

Virtualisierte Ausführungsplattform für die zuverlässige Steuerung cyber-physikalischer Systeme

2 Zeitraum

SoSe 2013 und WiSe 2013/14

3 Umfang

Jeweils 8 SWS

4 Veranstalter

Boguslaw Jablkowski, Informatik 12, OH-16, Raum 103, Tel. 6330, boguslaw.jablkowski@tu-dortmund.de
Markus Buschhoff, Informatik 12, OH-16, Raum 103, Tel. 6317, markus.buschhoff@tu-dortmund.de
Olaf Spinczyk, Informatik 12, OH-16, Raum E01, Tel. 6322, olaf.spinczyk@tu-dortmund.de

5 Aufgabe

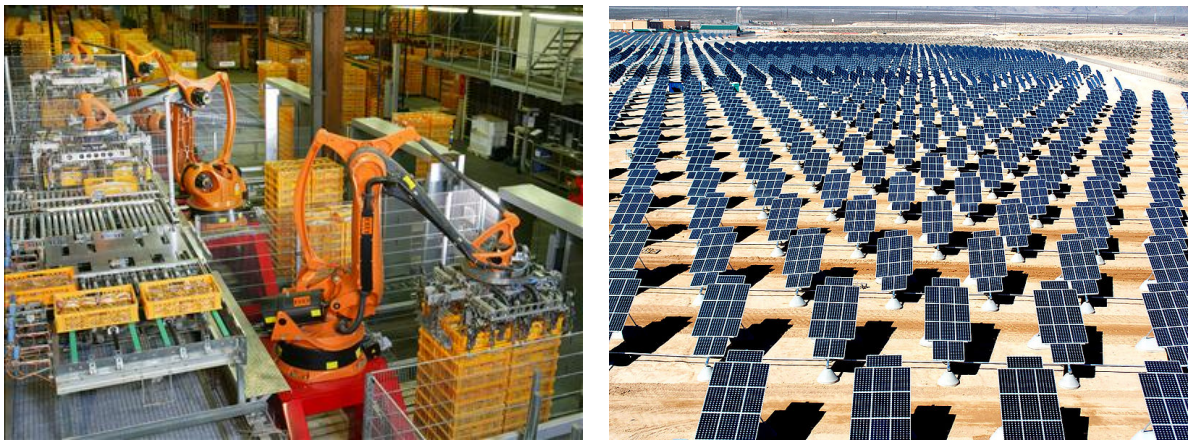

Abbildung 1: Cyber-physikalische Systeme finden zum Beispiel Anwendung in modernen Produktionsanlagen und Energienetzen. Sie verbinden eingebettete Systeme mit der physischen Umgebung.

5.1 Motivation

Cyber-physikalische Systeme werden in der Industrie für die Überwachung und Steuerung von großen und komplexen technischen Systemen eingesetzt und entstehen aus der Verschmelzung von vernetzten eingebetteten Rechensystemen mit mechanischen und elektronischen Elementen der physischen Umgebung. Beispiele für solche Infrastrukturen sind moderne Produktionsanlagen oder Energienetze. Um technische Neuerungen schneller

einführen zu können und den weiteren Automatisierungsprozess zu gewährleisten, sollen künftige physische Systeme verstärkt durch Software überwacht und gesteuert werden. Die steigende Bedeutung von Software und die wachsenden Anforderungen bezüglich der Erweiterbarkeit und Kosteneffizienz solcher Systeme erhöhen den Integrationsdruck. Eine etablierte Technik für Systemintegration ist die Virtualisierung [Hei08]. Diese erweitert herkömmliche Systeme um die Möglichkeit, Anwendungen an einem beliebigen Ort innerhalb einer virtualisierten Infrastruktur auszuführen. So ist es möglich, mehrere Anwendungen als verschiedene virtuelle Maschinen (Gast-Systeme) auf demselben physischen Rechner (Host-System) zu konsolidieren. Zudem bietet Virtualisierung zahlreiche Methoden zur Fehlertoleranz [BS96,RK07,CLM+08]. So kann z.B. durch Migration der Gast-Systeme zu einem anderen Host flexibel auf Hardware-Ausfälle reagiert werden. Virtualisierung erlaubt auch den einfachen Aufbau von Redundanzmechanismen, wie z.B. die Realisierung von „N-Version Programming" [AC77].

Um den zuverlässigen Betrieb von Überwachung- und Steuersoftwarekomponenten zu gewährleisten, müssen mindestens zwei wichtige Voraussetzungen erfüllt werden: die zugrunde liegende, virtualisierte Ausführungsplattform muss fehlertolerant und echtzeitfähig sein. Die Grundfrage, die sich folglich stellt, ist: Können mittels Virtualisierung diese notwendigen nichtfunktionalen Eigenschaften garantiert werden? Um in diesem Bereich Lösungen für die neuartigen Herausforderungen einer echtzeitfähigen Virtualisierungsinfrastruktur erforschen zu können, wird eine Laufzeitumgebung benötigt, welche die Ausbringung von echtzeitfähigen virtuellen Maschinen steuert, bei Bedarf Planungsalgorithmen zur Ressourcenverteilung anstößt (Scheduling) und eine Überwachung der virtualisierten Infrastruktur ermöglicht. Diese Plattform soll dadurch die Grundlage für die Evaluation der entworfenen Ansätze bilden.

Die Ergebnisse der Projektgruppe sind für eine aktuelle DFG-Forschergruppe 1511 an der TU Dortmund von Bedeutung. Zielsetzung der Forschergruppe ist es, Schutz- und Leitsysteme zur zuverlässigen und sicheren elektrischen Energieübertragung zu erforschen. Dabei sollen die dazu entworfenen Softwarekomponenten für die Schutz- und Steuerfunktionen in einer virtualisierten und echtzeitfähigen Umgebung laufen. Im Rahmen der Projektgruppe wird zur Verdeutlichung des praktischen Bezuges eine Exkursion zu einer Schaltanlage eines Energieversorgers angeboten.

5.2 Aufgabenstellung

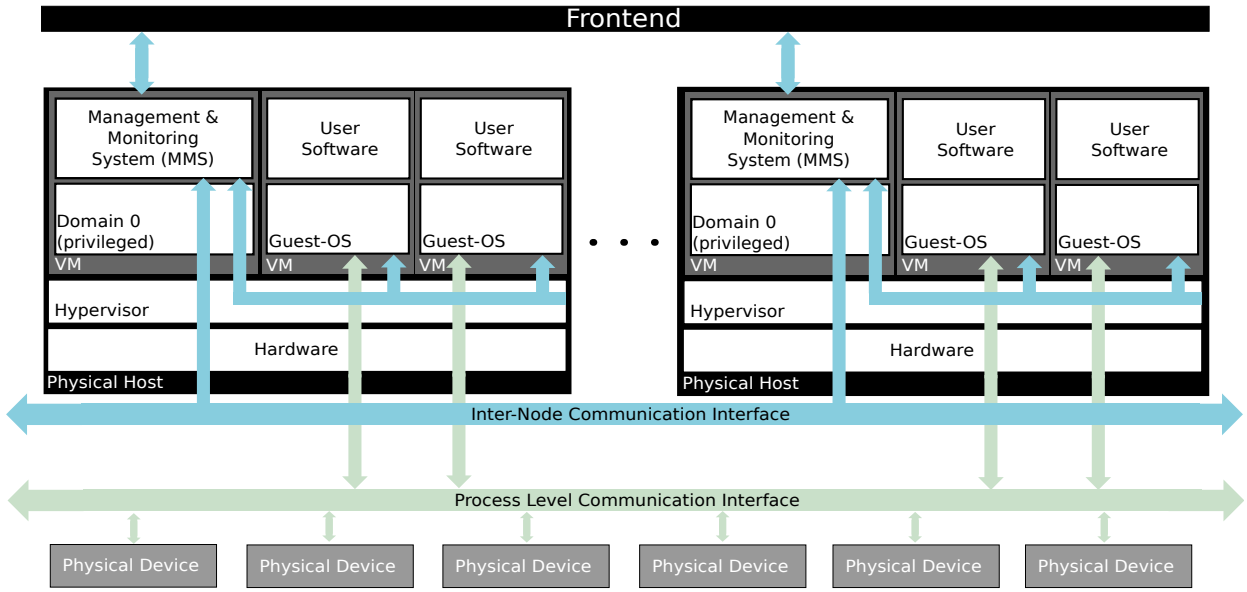

Abbildung 2: Beispielarchitektur

Die Projektgruppenteilnehmer sollen eine virtualisierte Ausführungsplattform für cyber-physikalische Anwendungen konzipieren, entwerfen und prototypisch implementieren. Dies soll anhand von erprobten Virtualisierungslösungen (z.B. XEN [BDF+03], Linux KVM) geschehen.

Die Plattform soll so umgesetzt werden, dass die im Weiteren angegebenen Anwendungs- und Managementanforderungen berücksichtigt werden.

Es müssen die nichtfunktionalen Eigenschaften, wie Echtzeitfähigkeit und Fehlertoleranz, durch die Ausführungsplattform garantiert werden, um die hohen Zeit- und Sicherheitsanforderungen der Softwarekomponenten zu erfüllen. Weiterhin soll die Plattform im laufenden Betrieb wartbar und erweiterbar sein (inklusive dem Abschalten, Zuschalten und Tauschen von Hardwarekomponenten). Die dynamische Rekonfigurierbarkeit

der Plattform, also auch die automatische Anpassung nach Fehlern, soll unter der Beibehaltung von Echtzeitfähigkeit und Fehlertoleranz gewährleistet sein. Unter Konfiguration wird in diesem Kontext die Verteilung der Last, also der Gastsysteme, auf die einzelnen Hostsysteme verstanden.

Die Plattform soll in der Lage sein, die gegebenen Fehlertoleranzmechanismen auf Basis der von der Virtualisierungssoftware bereitgestellten Techniken, wie z.B. redundante Ausführung und Migration, umzusetzen. Die Teilnehmer müssen dazu abschätzen, welche Methoden bereits zur Verfügung stehen, welche durch Erweiterung der bestehenden Software bereitgestellt werden können und welche neu zu implementieren sind. Die Effizienz von Fehlertoleranzmechanismen und den auszuführenden Softwarekomponenten wird stark durch die Wahl eines entsprechenden Betriebssystems beeinflusst. Deswegen sollen in der Ausführungsplattform anstatt Allzweck-Betriebssysteme, welche hohe Anforderungen an die Ressourcen haben, maßgeschneiderte Betriebssysteme zum Einsatz kommen.

Um im Fehlerfall korrekt reagieren zu können, muss die Fehlertoleranzlösung eine gültige Konfiguration des Gesamtsystems finden. Dazu müssen die Hostsysteme verteilt die Aufgabe der Ressourcenzuweisung (Scheduling-Problem) lösen und die anfallende Last aufteilen. Hierzu sind entsprechende Kommunikationsprotokolle und Programmierschnittstellen zu implementieren und insbesondere das Problem lösen, mögliche Konfigurationen proaktiv, also vor dem tatsächlichen Eintreten von Fehlern, zu bestimmen. Dies ist nötig, damit die gegebenen Echtzeitbedingungen eingehalten werden können. Um zu entscheiden, ob eine Konfiguration die Echtzeitanforderungen erfüllt, muss diese auf Verzögerungszeiten, verursacht durch Berechnungen und Kommunikation, untersucht werden. Eine Möglichkeit dazu bietet die *Modular Performance Analysis* mit *Real-Time Calculus* (MPA-RTC) [WTVL06]. MPA-RTC ist eine formale Technik zur Leistungsbewertung von verteilten Echtzeitsystemen – mit ihr können sichere, obere Schranken für die Latenzen in einem System berechnet werden.

Weiterhin soll es eine vernetzte Konfigurations- und Überwachungsoberfläche geben, welche mit den Host-Systemen kommuniziert. Diese Bedienoberfläche ist, inklusive der notwendigen Protokolle, zu spezifizieren und zu implementieren. Sie soll es ermöglichen, Anwendungen und deren Fehlertoleranz-Richtlinien zu konfigurieren, das gewünschte Echtzeitverhalten festzulegen und diese Plattformkonfiguration an die Host-Systeme zu übertragen, welche dann die gewählten Richtlinien umsetzen. Zudem soll die Bedienoberfläche in geeigneter Weise über wichtige Systemereignisse informieren und Systemprotokolle zugreifbar machen. Abbildung 2 veranschaulicht eine Beispielarchitektur einer virtualisierten Ausführungsplattform für cyber-physikalische Systeme.

6 Teilnahmevoraussetzungen

6.1 Voraussetzung

- Nachweisbare Kenntnisse aus dem Bereich der Vorlesung ,,Betriebssysteme”
- Nachweisbare Kenntnisse aus dem Bereich der Vorlesung ,,Eingebettete Systeme”
- Verständnis englischsprachiger Artikel und Handbücher

6.2 Wünschenswert

- Unix-/Linux-Kenntnisse
- Programmierkenntnisse C++
- Service Computing, Vorlesung oder Kenntnisse
- Nachweisbare Kenntnisse aus dem Bereich der Vorlesung ,,Rechnernetze und Verteilte Systeme”

7 Minimalziele

- Erstellen und kompilieren einer lauffähigen Installation von XEN auf mehreren Host-Systemen
- Ausführen von minimalen Gastsystemen (Mini-Images mit Betriebssystem CiAO und Beispielanwendung)
- Erstellen einer Infrastruktur, die Gastsysteme migrieren und parallel ausführen kann
- Entwurf und Implementierung eines Protokolls zur Steuerung und Überwachung der Host-Systeme
- Umsetzung einer grafischen Bedienoberfläche (GUI) zur Steuerung und Überwachung der Host-Systeme
- Entwurf und Implementierung eines Fehlererkennungskonzeptes (Ausfall von Host- und Gastsystem)

- Entwurf und Implementierung von grundlegenden Fehlertoleranzmechanismen (Neustart eines Gastsystems, Neustart auf Host mit geringster Auslastung)
- Entwurf einer Programmierschnittstelle (API) zur Einbindung verteilter Scheduling-Algorithmen auf den Host-Systemen
- Entwurf und Implementierung eines Protokolls zur Kommunikation zwischen Host-Systemen
- Grundlegende Evaluierung der erstellten Architektur
- Vollständige Dokumentation

8 Literatur

Literatur

[AC77] A. Avižienis and L. Chen. On the implementation of n-version programming for software fault tolerance during execution. In *Proceedings of the IEEE International Computer Software and Applications Conference*, pages 149–155, 1977.

[ALRL04] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *Dependable and Secure Computing, IEEE Transactions on*, 1(1):11 – 33, jan.-march 2004.

[ASR+10] Sherif Akoush, Ripduman Sohan, Andrew Rice, Andrew W. Moore, and Andy Hopper. Predicting the performance of virtual machine migration. In *Proceedings of the 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, MASCOTS '10, pages 37–46, Washington, DC, USA, 2010. IEEE Computer Society.

[BDF+03] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.*, 37(5):164–177, October 2003.

[BS96] Thomas C. Bressoud and Fred B. Schneider. Hypervisor-based fault tolerance. *ACM Trans. Comput. Syst.*, 14(1):80–107, February 1996.

[CLM+08] Brendan Cully, Geoffrey Lefebvre, Dutch Meyer, Mike Feeley, Norm Hutchinson, and Andrew Warfield. Remus: High availability via asynchronous virtual machine replication. In *In Proc. NSDI*, 2008.

[Hei08] Gernot Heiser. The role of virtualization in embedded systems. In *Proceedings of the 1st workshop on Isolation and integration in embedded systems*, IIES '08, pages 11–16, New York, NY, USA, 2008. ACM.

[JS12] Boguslaw Jablkowski and Olaf Spinczyk. Continuous performance analysis of fault-tolerant virtual machines. In *Proceedings of the 1st GI Workshop on Software-Based Methods for Robust Embedded Systems (SOBRES '12)*, Lecture Notes in Informatics. German Society of Informatics, September 2012. to appear.

[RK07] Hans P. Reiser and Rüdiger Kapitza. Hypervisor-based efficient proactive recovery. In *In Proceedings of the IEEE Symposium on Reliable Distributed Systems*, pages 83–92, 2007.

[WTVL06] Ernesto Wandeler, Lothar Thiele, Marcel Verhoef, and Paul Lieverse. System architecture evaluation using modular performance analysis: a case study. *Int. J. Softw. Tools Technol. Transf.*, 8(6):649–667, October 2006.

[XWLG11] Sisu Xi, Justin Wilson, Chenyang Lu, and Christopher Gill. Rt-xen: towards real-time hypervisor scheduling in xen. In *Proceedings of the ninth ACM international conference on Embedded software*, EMSOFT '11, pages 39–48, New York, NY, USA, 2011. ACM.

9 Rechtlicher Hinweis

Die Ergebnisse der Projektarbeit und die dabei erstellten Software sollen der Fakultät für Informatik und der DFG-Forschergruppe 1511 uneingeschränkt für Lehr- und Forschungszwecke zur freien Verfügung stehen. Darüber hinaus sind keine Einschränkungen der Verwertungsrechte an den Ergebnissen der Projektgruppe und keine Vertraulichkeitsvereinbarungen vorgesehen.

10 PG-Realisierung

Im Folgenden ist ein Vorschlag für eine Zeitplanung der beiden Projektsemester (Tabellen 1 und 2) aufgeführt. Der Projektgruppe wird Gelegenheit gegeben, selbst einen Projektzeitplan zu erstellen und nach diesem, in Abstimmung mit dem Veranstalter, die Projektarbeit durchzuführen.

Aufgabe	**Anfang**	**Ende**	**Wochen**
Einarbeitungsphase			
Einstieg in Literatur, Seminar	KW 15	KW 16	2
Ideensammlung, Umreißung des Grobkonzeptes	KW 17	KW 18	2
Erstellung des Projektplanes	KW 19	KW 19	1
Analyse- und Entwurfsphase			
Analyse der Anforderungen	KW 20	KW 20	1
Auswahl und Bewertung der Techniken	KW 21	KW 21	1
Ausarbeitung des Konzeptes	KW 22	KW 23	2
Entwurf der Architekturkomponenten:	KW 24	KW 28	5
Frontend			
Steuer-/Überwachungskomponente			
Protokolle			
Dokumentation (parallel)	KW 17	KW 29	13

Tabelle 1: Projektplan Sommersemester

Aufgabe	**Anfang**	**Ende**	**Wochen**
Umsetzungsphase			
Konsolidierung	KW 42	KW 42	1
Implementierung der Architekturkomponenten	KW 43	KW 2	10
Evaluationsphase			
Test und Bewertung des Gesamtsystems	KW 3	KW 5	3
Dokumentation (parallel)	KW 43	KW 6	14

Tabelle 2: Projektplan Wintersemester

Zur Einarbeitung der Teilnehmer wird zu Beginn des ersten Projektsemesters ein Wochenendseminar abgehalten, bei dem sich die Teilnehmer intensiv mit Themen aus den Bereichen Virtualisierung, Echtzeitsysteme, Fehlertoleranz, Scheduling und verteilte Algorithmen beschäftigen sollen. Zur Vorbereitung des Seminars erhalten die Teilnehmer vor der vorlesungsfreien Zeit entsprechende Vortragsthemen und Literaturhinweise.

In der weiteren Vorbereitungsphase sollen die Teilnehmer ein grobes Architekturkonzept erarbeiten und eine Zeitplanung für die Realisierung dieses Konzeptes aufstellen.

Während des ersten Projektsemesters sollen die Teilnehmer dann eine Arbeitsumgebung bestehend aus drei Virtualisierungs-Hosts erstellen. Weiterhin sollen Konzepte und Entwürfe erarbeitet werden. Hierbei handelt es sich insbesondere die Spezifikation von Protokollen und APIs. Während dieser Phase können die Teilnehmer die zuvor installierte Virtualisierungplattform nutzen, um Fragestellungen experimentell zu bearbeiten.

Im zweiten Semester sollen die Konzepte dann realisiert werden. Neben der Implementierung der Bedienoberfläche (die Teilnehmer sollen selbst evaluieren, welche Programmiersprache und Technologie sie hierfür einsetzen) sind insbesondere die spezifizierten Protokolle in die Arbeitsumgebung so zu integrieren, dass zum einen die Gastsysteme korrekt gesteuert werden, andererseits auch eine Erweiterung um neue Scheduling-Verfahren über eine gegebene Programmierschnittstelle möglich wird. Zum Abschluss soll das Gesamtsystem untersucht und bewertet werden. Dazu sollen die Teilnehmer mit wissenschaftlichen Kriterien evaluieren, wie sich die Plattform in den im Fehlermodell angenommen Situationen verhält.

11 Erweiterungsmöglichkeiten

Folgende Möglichkeiten zur Erweiterung der definierten Minimalziele sind vorstellbar:

1. Umsetzung und Evaluierung von Fehlertoleranzstrategien auf Basis von Hochverfügbarkeits-Lösungen wie z.B. Remus

2. Implementierung von redundanten Ausführungsstrategien und Fehlererkennung (N-Version-Programming, Voting, sequenzielle Ausführung)

3. Implementierung einer redundanten Speicherung der für die Fehlertoleranzmechanismen relevanten Daten (Logfiles, Plattformkonfigurationen)

4. Entwurf und Implementierung eines Konzeptes zur Protokollierung und Bewertung von Performance-Größen (CPU-Auslastung, Speicherverbrauch, Netzwerk-Traffic, etc.)

5. Herstellen einer Plattform-Distribution (Installationsmedium)

12 Beantragung von Ressourcen

Die notwendige Hard- und Software für CyPhyControl soll aus Lehrstuhlmitteln oder eventuell mit Hilfe eines QUEST-Antrages beschafft werden. Die Gruppenarbeitstermine können im Laborraum der Arbeitsgruppe stattfinden. Lediglich für die Seminartermine und Gruppen- sowie Plenarsitzungen besteht zusätzlicher Raumbedarf. Die üblichen Mittel für Druck- und Vervielfältigungskosten werden hiermit beantragt.

13 Lehrdeputat

Olaf Spinczyk: 2 SWS
Boguslaw Jablkowski: 3 SWS
Markus Buschhoff: 3 SWS