Technologien und Konzepte zur autonomen Verwaltung von IT-Systemen

Gennadi Rabinovitch
Friedrich-Schiller-Universität Jena
Lehrstuhl für Datenbanken und Informationssysteme
Ernst-Abbe-Platz 2
07743 Jena
gennadi@informatik.uni-jena.de

Zusammenfassung

Die Computer-Industrie hat Jahrzehnte damit verbracht, immer komplexere Systeme zu entwickeln. Heute jedoch ist die Komplexität als limitierender Faktor in der Systemweiterentwicklung selbst das Problem. Moderne IT-Landschaften zeichnen sich durch eine heterogene Komponenten-, Architektur- und Applikationsvielfalt aus. Durch bloße menschliche Intervention sind diese kaum noch manuell kontrollier- und handhabbar. Neben einer vernünftigen Ressourcenplanung, Installation und Vorabkonfiguration sind die Bedienung, Administration und Optimierung im laufenden Betrieb unabdingbar, um Verfügbarkeit, Performance und vorgegebene Richtlinien einzuhalten.

Ansätze wie das Autonomic Computing versuchen die Bedienung und Administration von IT-Systemen zu vereinfachen, indem sie die komplexen Verwaltungs- und Konfigurationsaufgaben automatisieren und an die Systeme selbst übertragen. Dadurch wird es Unternehmen möglich, mit dem gleichen Personal auch erweiterte Systeme managen und sich auf die strategische, geschäftsorientierte Langzeitplanung konzentrieren zu können. Im vorliegenden Beitrag werden die Probleme der manuellen Systemverwaltung diskutiert und anschließend ausgewählte Konzepte und Technologien zur autonomen Systemadministration vorgestellt. Dabei werden die Aspekte der Autonomie im Datenbank-Performance-Umfeld in die Betrachtungen einbezogen.

1 Einleitung

In den letzten Jahrzehnten wurden immer leistungsfähigere und zunehmend komplexe Systeme entwickelt. Heute jedoch ist die Komplexität als limitierender Faktor in der Systemweiterentwicklung selbst das Problem. Die modernen IT-Systeme sind aufgrund ihrer heterogenen Komponenten-, Architektur- und Applikationsvielfalt sehr kostenintensiv in der Entwicklung, kaum noch manuell kontrollier- und handhabbar und dadurch überaus fehleranfällig. Neben einer vernünftigen Ressourcenplanung, Installation und Vorabkonfiguration sind die Bedienung, Administration und Optimierung im laufenden Betrieb unabdingbar, um Verfügbarkeit, Performance und vorgegebene Richtlinien einzuhalten.

Um die Administration von IT-Systemen zu vereinfachen, werden diese zunehmend um Komponenten erweitert, die eine automatisierte Systemüberwachung und -verwaltung unterstützen. Ansätze wie das Autonomic Computing helfen der steigenden Komplexität entgegen zu wirken, indem sie Technik nutzen, um Technik zu verwalten [IBM04]. Vorbild des Autonomic Computing ist das vegetative Nervensystem des Menschen. Ähnliches sollen autonome Computersysteme künftig leisten: Analog zum menschlichen Organismus werden alle wichtigen Parameter „unwillkürlich“ überwacht und es wird bei Abweichungen mit den richtigen Maßnahmen reagiert [GC03]. Das (Langzeit-)Ziel des Autonomic Computing ist die Selbstregulation von Computersystemen, die flexibel auf sich ändernde Arbeits- und Umgebungsbedingungen reagieren sollen, ohne ständige Eingriffe menschlicher Administratoren zu erfordern.

2 Administration heute

Mit der zunehmenden Komplexität und Leistungsfähigkeit von IT-Systemen steigen auch die Anforderungen an diese. Für viele Unternehmen hat der fehlerfreie Betrieb ihrer IT-Systeme allerhöchste Priorität, da sich Ausfälle oder Leistungseinbußen unmittelbar auf die finanzielle Situation des Unternehmens auswirken. In solchen Fällen müssen Probleme schnellstmöglich erkannt und behoben werden. Dazu müssen die Administratoren schnell und kompetent handeln. Allerdings sind sie nicht rund um die Uhr verfügbar, um etwa auf Performanceeinbußen sofort reagieren zu können. Durch mangelnde Kompetenz und Erfahrung kann es zusätzlich zu Zeitverzögerungen kommen, bis die Lösung gefunden werden kann.

Manuelle Administration ist nicht nur zeitaufwendig, kostspielig und fehleranfällig, sondern auch in den meisten Fällen nur noch durch erfahrene und gut ausgebildete Spezialisten durchführbar. Die Anforderungen an die Kompetenz, Anzahl und Lernfähigkeit der Fachkräfte und damit die laufenden Betriebskosten steigen somit beträchtlich.

Die immer komplexer werdenden Systeme bestehen aus einer Vielzahl von Komponenten mit diversen Abhängigkeiten zwischen den und innerhalb der Komponenten selbst. All diese Komponenten müssen überwacht und verwaltet werden. Da einige Komponenten auf den gleichen Ressourcen operieren, kann Tuning einer den Rückgang in Performance einer anderen Komponente verursachen. Solche Abhängigkeiten sollen automatisch erkannt und berücksichtigt werden.

Bei der Konfiguration der IT-Systeme ist auch die Berücksichtigung ihrer Dynamik unabdingbar. Beim Verändern der Arbeitslast müssen gewisse Parameter verändert oder sogar u. U. neue Komponenten hinzugeschaltet bzw. entkoppelt werden, was die Verwaltung und Optimierung von IT-Systemen zu einem iterativen Prozess macht. Durch eine periodische Überwachung der Systemdynamik und eine automatische Ausführung vordefinierter Aktionen (wie Rekonfiguration bzw. Ressourcenreallokation) kann das Problem der Systemdynamik adressiert werden.

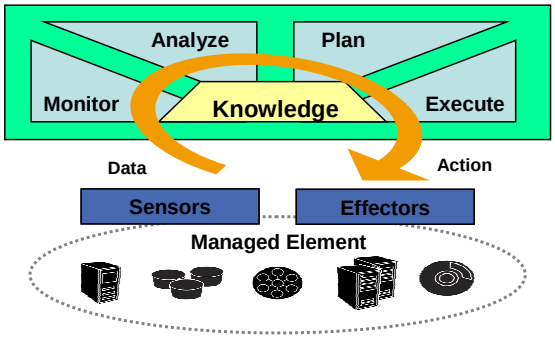
3 Vision der autonomen Datenbankadministration

Das Ziel vieler aktueller Forschungsansätze ist es, selbst verwaltende IT-Systeme zu entwickeln, die durch den Administrator auf abstrakter Ebene formulierte Richtlinien (auch bekannt als Service Level Agreements) autonom zu erreichen versuchen. Die Administratoren können dadurch von der alltäglichen, fehleranfälligen Arbeit entlastet werden und bekommen die Gelegenheit, sich auf die strategische, geschäftsorientierte Langzeitplanung zu konzentrieren.

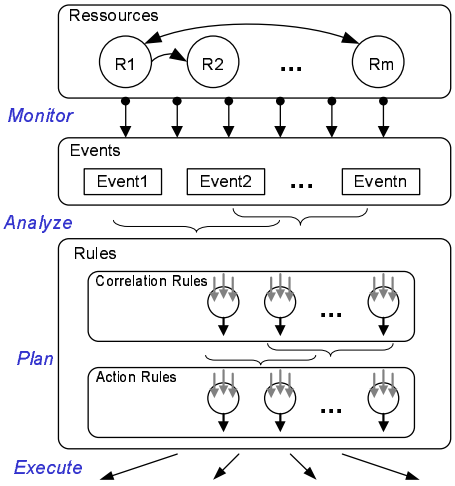
Ein *ideales*, selbstverwaltendes Datenbanksystem würde also u. a. die aktuelle Performance überwachen und das Systemwissen dafür verwenden, diese zu verbessern; nach Bedarf entsprechende Parameter dynamisch zur Laufzeit verändern können; je nach Workload die Speicherauslastung erhöhen bzw. senken; automatisch die Workloadtypen erkennen und entsprechende Indexe vorschlagen bzw. erstellen oder sogar die Datenbank partitionieren; automatisch den Bedarf der Wartungsoperationen (wie REORG, Sammeln von Statistiken, Backup, Load bzw. Rebind) bestimmen, ensprechend die voraussichtliche Dauer abschätzen und diese Operationen ausführen; nach Möglichkeit Probleme vorhersagen und entsprechende vorbeugend-korrigierende Aktionen vorschlagen bzw. vornehmen, um diese zu vermeiden.

4 Konzepte und Technologien zur Automatisierung der Datenbankadministration

Die vier Hauptaspekte der Selbstverwaltung sind Selbstkonfiguration, Selbstheilung, Selbstoptimierung und Selbstschutz [KC03]. Im Folgenden werden Konzepte und Technologien vorgestellt,



(a) Das MAPE-Paradigma



(b) Correlation Engine

Abbildung 1: Die MAPE-Schleife und ihre Umsetzung mit einer Correlation Engine

die eine Erstellung (ansatzweise) selbstverwaltender Datenbanksysteme ermöglichen.

4.1 Das MAPE-Paradigma

Ein autonomes Computersystem basiert typischerweise auf dem MAPE-Paradigma [IBM04]. Die Idee dahinter ist aus der Feedback-Kontrolltheorie [Hel04] hinreichend bekannt: Die Komponenten des Systems werden ständig überwacht und bei bestimmten Zustandsänderungen werden Aktionen initiiert, die den Zustand der Komponenten wiederherstellen sollen; diese verursachen wiederum Zustandsänderungen, die überwacht werden usw.

Die im Folgenden dargestellten vier Phasen des MAPE-Paradigmas bilden einen (potentiell endlosen) iterativen Prozess (Abbildung 1(a)). Die Phasen können dabei durch einzelne Komponenten realisiert werden, die aufeinander aufbauen und untereinander Informationen austauschen.

- **Monitor:** Die Monitor-Komponente überwacht die Ressourcen über die sog. Sensoren[1] und sammelt Daten, die je nach Anwendung detaillierte Informationen über die Ressourcenkonfiguration, -status, -kapazität und -durchsatz sowie Topologieinformationen, Metriken etc. enthalten können. Diese Daten werden vom Monitor gespeichert, aggregiert, korreliert und gefiltert, bis die Gesundheit des Systems gefährdende Symptome identifiziert werden können, die anschließend von der Analyze-Komponente ausgewertet werden.

- **Analyze:** Der Analyzer bietet Mechanismen, um Symptome zu korrelieren und die über die möglichen Probleme getroffenen Annahmen zu überprüfen. Mit Hilfe dieser Komponente kann das System seine Umgebung erforschen, um Vorhersagen über das künftige Systemverhalten zu treffen.

- **Plan:** Die Planungsfunktionalität ermittelt die Aktionen, die zum Erreichen der vorgegebenen Zielrichtlinien ausgeführt werden müssen. Bei der Planung werden die Systemkomponenten mit allen Abhängigkeiten in Betrachtungen einbezogen, um das System stabil zu halten und Oszillationen[2] zu vermeiden.

[1] Ein Sensor ist ein Teil der Ressourcen-Schnittstelle, der für den *lesenden* Zugriff auf die Ressourcen-Informationen verantwortlich ist.

[2] Insbesondere soll eine Oszillation zwischen zwei suboptimalen Zuständen vermieden werden.

- **Execute:** Die in der Plannungsphase ermittelten Aktionsabfolgen werden anschließend durch die Execute-Komponente mit Hilfe von Effektoren[3] der Ressourcen ausgeführt. Die somit verursachten Systemänderungen werden wiederum durch den Monitor überwacht.

Um autonom agieren zu können, benötigt ein autonomes System Wissen über die Anforderungen an das System, über seine Infrastruktur sowie über die Abhängigkeiten unter den einzelnen Komponenten. Dieses Wissen kann in drei Bereiche eingeteilt werden: (1) das Wissen über die Topologien, (2) das Wissen über die Richtlinien und (3) das Wissen, welches zur Lösung von identifizierten Problemen benötigt wird.

4.2 Correlation Engine

Correlation Engines [MS03] sind autonome Komponenten, die eine fortlaufende, automatisierte Analyse von Echtzeitevents basierend auf nutzerdefinierten, konfigurierbaren Regeln durchführen. Mit Hilfe von diesen Regeln können Systemversagen, Performanceprobleme bzw. komplexe Angriffsmuster erkannt und entsprechende Aktionen initiiert werden [SSM+04].

Ein Correlation-Engine-Modell besteht typischerweise aus drei Schichten: einer Ressourcen-, einer Event- und einer Regel-Schicht (Abbildung 1(b)). Die Ressourcen-Schicht wird durch ein Ressourcen-Modell repräsentiert, welches zur Beschreibung des zu verwaltenden Systems von zentraler Bedeutung ist. In diesem Modell werden die Abhängigkeiten unter den einzelnen Ressourcen, die sowohl physische Hardware- als auch logische Software-Komponenten oder gar virtuelle Dienste sein können, beschrieben. Events in der Event-Schicht repräsentieren signifikante Änderungen des Ressourcen-Zustands. Diese Events werden durch die Correlation Rules analysiert und gefiltert, indem beispielsweise mehrere einzelne Events zu einem kontextrelevanten Event zusammengefasst werden. Action Rules werten anschließend die durch Correlation Rules generierten Events aus und lösen entsprechende korrigierende Aktionen aus bzw. sammeln weitere Informationen über die überwachten Ressourcen, um beispielsweise das eingetretene Problem einzugrenzen.

Mit Hilfe von Correlation Engines kann das MAPE-Paradigma (Abschnitt 4.1) umgesetzt werden. In Abbildung 1(b) wird die Zuordnung der einzelnen Phasen der MAPE-Kontrollschleife zu den Schichten der Correlation Engine aufgezeigt.

4.3 Weitere Technologien

In diesem Abschnitt werden weitere ausgewählte Technologien zur Unterstützung der autonomen Administration kurz vorgestellt. Mit jeder der hier eingeführten Technologien kann das MAPE-Paradigma ähnlich wie mit der Correlation Engine (Abschnitt 4.2) umgesetzt werden.

- **ACT:** Das IBM Autonomic Computing Toolkit ist eine Sammlung von Technologien, Werkzeugen, Beispielen, Szenarien und Dokumentationen, die dem Nutzer die Möglichkeit geben, autonomes Verhalten in ihre Systeme einzubauen [Cyb05]. Mit Hilfe des ACT können Systeme um Selbstheilungs- und Selbstoptimierungseigenschaften erweitert werden.
- **PMAC:** Das Policy Management for Autonomic Computing ist eine Infrastuktur, die in Softwaresysteme eingebunden werden kann, um ihre Verwaltung zu vereinfachen. Diese Infrastruktur besteht aus einem Autonomic Manager, der anhand von Regeln Entscheidungen treffen kann, einer Komponente zum Speichern von Policies, Nutzerbibliotheken und APIs [IBM05].

[3] Im Gegensatz zu Sensoren sind Effektoren für *schreibenden* Zugriff auf die Ressourcen-Parameter verantwortlich.

- **TEC:** Die IBM Tivoli Enterprise Console ist eine regelbasierte Event-Management-Anwendung, die System-, Netzwerk-, Datenbank- und Anwendungsverwaltung integriert, um die Verfügbarkeit einer Ressource zu erhöhen [IBM02].

5 Zusammenfassung und Ausblick

Die zunehmende Komplexität der IT-Systeme macht diese kaum noch durch bloße menschliche Intervention kontrollier- und handhabbar. Die Administration dieser Systeme wird immer aufwendiger, kostspieliger und fehleranfälliger. Ansätze wie das Autonomic Computing wirken der zunehmenden Komplexität entgegen, in dem sie die komplexen Verwaltungs- und Konfigurationsaufgaben automatisieren und an die Systeme selbst zu übertragen versuchen. In diesem Beitrag wurden die Probleme der manuellen Systemverwaltung sowie einige Lösungsansätze diskutiert. Es wurde das MAPE-Paradigma, welches dem Autonomic Computing zu Grunde liegt, vorgestellt. Weiterhin wurde eine Referenzarchitektur von Correlation Engines vorgestellt und es wurde skizziert, wie mit einer Correlation Engine die MAPE-Kontrollschleife umgesetzt werden kann. Im Anschluss wurden ausgewählte IBM-Technologien vorgestellt, die zur Unterstützung der autonomen Systemadministration benutzt werden können.

Im Rahmen eines Drittmittelprojekts wird am Lehrstuhl für Datenbanken und Informationssysteme eine Infrastruktur zur Automatisierung typischer Datenbank-Tuning-Aufgaben entwickelt. Dabei stehen sowohl das Formalisieren, Speichern, Verwalten, Austauschen und individuelle Anpassen, als auch das automatische ereignisgesteuerte Anwenden dieses Tuning-Wissens im Vordergrund. Eine zentrale Instanz überwacht dabei ständig den Systemzustand auf akkute und bevorstehende Probleme und kann bei Performance-Verschlechterungen korrigierende Aktionen anstoßen.

Literatur

[Cyb05] J. R. Cybrynski. *ABCs of the Autonomic Computing Toolkit.* IBM, September 2005.

[GC03] A. G. Ganek und T. A. Corbi. The Dawning of the Autonomic Computing Era. *IBM Systems Journal*, Vol. 42(1), 2003.

[Hel04] J. L. Hellerstein. Self-Managing Systems: A Control Theory Foundation. *29th Annual IEEE International Conference on Local Computer Networks (LCN'04)*, 2004.

[IBM02] IBM. *IBM Tivoli Risk Manager 3.8 and the TEC Database. Redbook*, 2002.

[IBM04] IBM. *An Architectural Blueprint for Autonomic Computing. White Paper*, 2004.

[IBM05] IBM. *Policy Management for Autonomic Computing. Developer's Guide and Reference. Version 1.2.1*, Nov 2005.

[KC03] J. O. Kephart und D. M. Chess. The Vision of Autonomic Computing. *Computer*, 36(1):41–50, 2003.

[MS03] A. Maedche und L. Stojanovic. IBM Correlation Engines. Technical Report 2-8-04/03. *FZI Research Center for Information Technologies, University of Karlsruhe*, 2003.

[SSM+04] L. Stojanovic, J. Schneider, A. Maedche et al. The Role of Ontologies in Autonomic Computing Systems. *IBM Systems Journal*, Vol. 43(3), August 2004.