Vorverarbeitung von dynamischen Prozessdaten für proaktives Störungsmanagement

Exposé zur Diplomarbeit

zur Erlangung des akademischen Grades
Diplominformatiker

HUMBOLDT-UNIVERSITÄT ZU BERLIN
MATHEMATISCH-NATURWISSENSCHAFTLICHE FAKULTÄT II
INSTITUT FÜR INFORMATIK

eingereicht von:	Moritz Brettschneider
geboren am:	08.09.1986
in:	Berlin
Gutachter:	Prof. Dr. Leser
	Uwe Toman

1 Motivation

Computersysteme sind aus dem alltäglichen Leben, wie aus der Produktion nicht mehr wegzudenken. Hochkomplexe digitale Abläufe steuern große Bereiche der Industrie.

Dabei gibt es praktisch kein System, das vollkommen fehlerfrei seine Arbeit verrichtet. Fehler können unterschiedliche Auswirkungen haben und bis zum Ausfall ganzer Produktionsprozesse führen, was große finanzielle Einbußen sowie einen Verlust an Produktivität bedeuten kann[LGLS07].

Eine große Hilfe wäre es, drohende Ausfälle von Systemen im Vorhinein angedeutet zu bekommen, um eventuelle Ursachen vor dem Ausfall beseitigen zu können.

In modernen Computersystemen werden Logsysteme genutzt, die große Mengen an Logfiles erzeugen. Gespeichert werden zahlreiche Ereignisse, vor allem aber auch Fehlermeldungen. Fehlermeldungen deuten Systemausfälle oft vor dem Ausfall an und können so genutzt werden, um proaktiv in das System einzugreifen und einen drohenden Ausfall zu verhindern. Die hohe Komplexität von Computersystemen spiegelt sich jedoch auch in deren Logfiles wieder, welche auf mehrere Gigabyte anwachsen können[ZLPG09]. Durch manuelles Beobachten ist es daher nicht möglich, mit angemessenem Aufwand Probleme im Vorfeld zu erkennen.

Fehlermeldungen treten oft in ähnlichen Abfolgen auf, sodass Fehlerabläufe die zu Ausfällen führen von maschinell angelernten Modellen erkannt werden und entsprechende Warnmeldungen ausgegeben werden können.

Logfiles werden jedoch nicht in strukturierter Form erzeugt, sondern in Klartext oder semi-strukturierter Form und enthalten viele redundante Informationen[ZLPG09].

Um die Nutzung von nichtstrukturierten Logfiles für die Vorhersage von Systemausfällen zu ermöglichen und zu optimieren, ist es notwendig die Logdaten einer Vorverarbeitung zu unterziehen[ZLPG09].

2 Zielstellung

In der Diplomarbeit soll eine geeignete Form für die Vorverarbeitung von Logdateien gefunden werden, um die enthaltenen Ereignisse automatisch für die maschinelle Verarbeitung zu optimieren und für die weitere Analyse effektiv nutzbar zu machen.

Dazu wird zunächst die typische Ausgangslage in den betreffenden Produktionsprozessen identifiziert und in diesem Kontext die typischen Datenstrukturen und deren Eigenschaften analysiert.

Zentrale Aufgabe besteht in der Entwicklung eines geeigneten Vorgehensmodells zur effizienten Aufbereitung der semi-strukturierten Logfiles und deren größtmögliche Umsetzung mittels Splunk[So11] als Vorbereitung der Datenanalyse. Auch die Vorhersage von Systemausfällen soll prototypisch umgesetzt werden.

Zusätzlich werden die bisherigen Vorgehensweisen und deren Schwachstellen identifiziert und mit dem vorgeschlagenen automatisierten Verfahren verglichen.

3 Vorgehen

Das Vorgehen orientiert sich wesentlich an dem von Felix Salfner vorgeschlagenen Ablauf in [SaTs08] für die Nutzung eines Hidden semi-Markov Model (HSMM). Nicht bei allen Vorhersagealgorithmen werden alle im Folgenden beschriebenen Schritte bis ins letzte Stadium genutzt.

In der Diplomarbeit wird ein modulares System Gegenstand der Untersuchungen sein. Die Ursachen für Ausfälle können in Teilsystemen zu finden sein, aber durch gegenseitige Beeinflussung auch zu Fehlermeldungen in anderen Subsystemen führen. Logdaten werden verschiedenen Charakter besitzen und verschiedene Strukturen aufweisen. Eine zentrale Aufgabe wird daher zunächst die Analyse der vorhandenen Logdaten sein, um deren spezifische Eigenschaften in Erfahrung zu bringen. Aus dieser Analyse müssen Schlussfolgerungen für fast alle kommenden Verarbeitungsschritte geschlossen werden.

Grenzen zwischen Logeinträgen erkennen Einzelne Logmeldungen können über mehrere Zeilen gehen. Um auf der Ebene von Logeinträgen rechnergestützt zu analysieren, müssen die Grenzen erkannt und für die weitere Verarbeitung markiert werden.

Fehler-IDs erstellen Fehleranalysewerkzeuge benutzen oft ganze Zahlen um den Typen eines Fehlers zu charakterisieren[SaTs08]. Daher werden in diesem Schritt die Meldungen so verarbeitet und reduziert, dass möglichst nur die Typen der Fehler in den resultierenden Strings widergespiegelt werden.
Zuerst werden dazu wie in [SSSZ04] alle Meldungen aus dem Log entfernt, welche einen Informationscharakter haben, also keinem Fehler entspringen. Dann werden logeintragspezifische Daten, wie z.B. Nummern, Zeitstempel und IP-Adressen entfernt.

Nagappan und Vouk[NaVo10] erzeugen eine Tabelle mit den Vorkommenshäufigkeiten aller im Logfile enthaltenen Wörter, um diese für die Analyse von variablen bzw. statischen Teilen einer Lognachricht zu nutzen. Dies könnte genutzt werden um variable Teile zu entfernen und so die Redundanz der geloggten Informationen weiter zu verringern.

Letztlich werden Einträge zusammengefasst, die eine geringe Levenshtein-Distanz aufweisen und diesen gemeinsam eine Fehler-ID zugewiesen. Der Zeitstempel wird extrahiert und zur weiteren Nutzung bereitgestellt, wobei zu beachten ist, dass verschiedene Logsysteme unterschiedliche Zeitstempelformate benutzen.

Zusammenführen von Logdateien Die bisherigen Schritte müssen für die einzelnen Systeme getrennt voneinander durchgeführt werden. Für die weitere Verarbeitung werden die Logdaten zusammengeführt. Dabei wird die Herkunft der Loginformationen konserviert.

Erstellen von Fehlertupel Fehlermeldungen, die mehrmals zur gleichen Zeit auftreten, gehören oft zur selben Fehlerursache[LZSSMG05, SaTs08]. Das Anwenden von Tupling führte in [Bu96] zu einer bedeutenden Reduzierung des Volumens der Meldungen. Wie in [LZSSMG05, SaTs08] werden wir Fehlermeldungen in Tupel zusammenfassen, die innerhalb einer bestimmten Zeit ε auftreten. Das optimale ε wird wie in [TsSi83] durch Plotten der Anzahl der Tupel über ε und das Feststellen des Wertes am Knick der entstandenen L-Kurve gewählt.

Extraktion von Logsequenzen Im dritten Schritt werden Logsequenzen extrahiert. Dabei wird in Ausfall- und Nichtausfallsequenzen unterschieden. Diese werden für das Training von Klassifikatoren benötigt. Bei den Ausfallsequenzen werden Sequenzen von Fehlermeldungen erzeugt, an deren Ende ein Ausfall folgte. Nichtausfallsequenzen werden so gewählt, dass sie einen bestimmten Abstand zu Ausfällen besitzen und damit ein Zusammenhang unwahrscheinlich ist. Bei der Extraktion von Sequenzen sind die folgenden Parameter sinnvoll zu wählen[SaTs08]:

- *Vorlaufzeit (Lead-time)* Δt_l, die Zeitspanne zwischen Fehlersequenzen und dem Ausfall
- *Fensterlänge* Δt_d, die maximale Länge der Sequenzen
- *Abstand für Nichtausfallsequenzen* Δt_m, die Zeitdauer die zwischen einer Nichtausfallsequenz und einem Ausfall liegen darf

Clusteranalyse der Fehlersequenzen Anschließend werden die Fehlersequenzen so zu Clustern zusammengefasst, dass gleiche Fehlermechanismen in der gleichen Klasse gesammelt werden. Um eine Abstandsmatrix zu erhalten, trainiert Salfner in [Sa08] kleine HSMMs für jede der Fehlersequenzen und berechnet dann jeweils für alle Sequenzen die Sequenzwahrscheinlichkeiten.

Da die Anzahl der Klassen im Vorhinein nicht bekannt ist, bieten sich für die Clusteranalyse hierarchische Verfahren an. Eine geeignetes Verfahren wird hier gewählt werden.

Rauschen entfernen Bei der Clusteranalyse entstehen Fehler, die durch Rauschen in den Logfiles bedingt sind[Sa08]. Diese Fehler sollen durch entsprechende Filter entfernt werden und die Klassen so noch präziser werden.

Es werden dabei die Häufigkeiten des Auftretens der verschiedenen Ereignisse in den Sequenzen analysiert und nur solche beibehalten, die in den Klassen signifikant öfter auftreten[Sa08].

Vorverarbeitung in der Vorhersage Um Ausfälle des Systems vorauszusagen, müssen die bisher beschriebenen Schritte auf die jeweils aktuellen Logdaten in gleicher Weise angewandt werden.

Zu beachten ist, dass nicht alle Fehler im Vorhinein bekannt sein können. Laufende Systeme werden durch Patches oder Erweiterungen verändert, was dazu führen kann, dass unbekannte Fehlermeldungen auftreten. Hier muss ein geeigneter Umgang gefunden werden.

Klassifikation Ist die Vorverarbeitung der Logdaten abgeschlossen, können die entstandenen Klassen für das Anlernen von Klassifikatoren genutzt werden.

Um die Vorverarbeitung zu Testen sollen zwei verschiedene Modelle genutzt werden.

Salfner schlägt vor oder nutzt in seinen Arbeiten [Sa05, Sa06, SaTs08] für die Klassifikation Hidden-Markov-Modelle oder Varianten. Auch in dieser Arbeit soll ein Hidden Markov Model (HMM) oder eine Variante genutzt werden.

Fulp, Fink und Haack[FuFiHa08] nutzten lediglich beim Loggen erzeugte Fehler-IDs für die Klassifikation einer Support Vector Machine (SVM). Hier könnte man aber auch die anfangs bei der Clusteranalyse der Fehlermeldungen erzeugte ID nutzen. Für die Klassifikation wurden sowohl aggregierte Informationen als auch Sequenzinformationen in einen Vektor geschrieben. Die aggregierten Informationen bestehen aus Tupeln von Fehler-ID und der Häufigkeit des Auftretens dieses in einem bestimmten Zeitfenster. Für die Sequenzinformationen werden die Fehler-IDs gruppiert. Die Sequenz der Gruppenzugehörigkeit der Elemente innerhalb einer bestimmten Fensterlänge wird gespeichert. Das Auftreten der verschiedenen Sequenzen wird, wie bei den aggregierten Informationen, in Tupeln von Sequenz und Häufigkeit des Auftretens der Sequenz gespeichert. Die so gebildeten Tupel können als Vektoren für das Anlernen einer SVM und bei der Vorhersage für die Klassifikation genutzt werden. In der Diplomarbeit soll auch eine SVM in dieser Weise genutzt und getestet werden.

Evaluation Die Clusteranalyse zur Vergabe von Fehler-IDs wird anhand eines Ähnlichkeitsplots der gegenseitigen Levenshtein-Distanzen einiger zufällig gewählter, vorverarbeiteter und sortierter Fehlermeldungen wie in [SaTs08] evaluiert. Auch werden die Klassen durch Stichproben manuell überprüft. Eine weitere Kennzahl ist der Grad der Reduzierung der Fehlereinheiten für die weitere Verarbeitung.

Bei der Erstellung der Tupel soll eine weitere Reduzierung der Fehlereinheiten bei Beibehaltung der Informationen erzielt werden. Hier kann die Reduzierung gemessen werden. Außerdem kann die Klassifikation mit und ohne Tupling durchgeführt und die Genauigkeit bei der Vorhersage verglichen werden. Auch hier können Stichproben einen Einblick in die Plausibilität der Zusammenfassung der Meldungen liefern.

Die Homogenität der Sequenzklassen wird vor und nach der Rauschentfernung manuell überprüft.

Die Vorverarbeitung wird außerdem durch die Klassifikation, von beim Training unbekannten Logdaten, überprüft. Dies wird durch n-fold-Tests organisiert. Wichtige Metriken werden in diesem Zusammenhang Precision, Recall, F-Maß, Receiver Operating Characteristic (ROC) und Area under the curve (AUC) darstellen.

4 Verwandte Arbeiten

Salfner implementierte und untersuchte in seiner Dissertation[Sa08] einen Online-Algorithmus zur Vorhersage von Ausfällen in einem Telekommunikationssystem nach dem oben beschriebenen Ablauf. In [SaTs08] zeigte er, wie die Vorhersagegenauigkeit von der Vorverarbeitung abhängt. Dabei wurden im F-Maß bei Auslassen des Clusterns der Sequenzen nur 77% des Ergebnisses mit Gruppierung der Sequenzen, beim Wegfall der Rauschentfernung gar nur 55% des Ergebnisses mit Rauschentfernung erreicht. In [Sa08] wird bemerkt, dass 25% der Ausfälle im Vorhinein erkannt werden konnten.

Fulp, Fink und Haack[FuFiHa08] erreichten in Experimenten mit dem oben beschriebenen Ablauf bei der Vorhersage von Festplattenausfällen in einem Computercluster Genauigkeiten von bis zu 73% bei einer Vorlaufzeit von zwei Tagen.

Liang et al.[LZSSMG05] reduzierte die Anzahl der Einträge in den Logs von einem 8'192 Prozessor BlueGene/L Prototypen um 99,96% in drei Schritten. Der Korpus bestand im Rohzustand aus 828'387 Meldungen. Aus diesen wurden zuerst alle Meldungen entfernt, die weder als FATAL noch als FAILURE markiert wurden. Dann wurden alle Einträge entfernt, welche von einer Konsole aus abgeschossen wurden, da sie weder einen Hard- noch Softwarefehler darstellen. Anschließend wurden die Fehler kategorisiert. Sie beobachteten, dass Fehler an bestimmten Komponenten explosionsartig in wenigen Sekunden Abstand

auftraten. Diese bezeichnen sie als Cluster. Bei Anwendung eines Fensters von zwei Minuten, wurden im zweiten Schritt die Fehlermeldungen auf 9150 Cluster reduziert. In einem dritten Schritt wird über die Komponenten hinweg in Stundenabschnitten zusammengefasst. Dabei wurden die Cluster auf 304 Einheiten reduziert.

Literatur

[Bu96] Michael Francis Buckley, Daniel P. Siewiorek; A comparative analysis of event tupling schemes, In: *FTCS-26, Intl. Symp. on Fault Tolerant Computing*, 294303, 1996.

[FuFiHa08] Errin W. Fulp, Glenn A. Fink, Jereme N. Haack; Predicting computer system failures using support vector machines, In: *Proceedings of the First USENIX conference on Analysis of system logs*, USENIX Association, 2008.

[LGLS07] Yawei Li, Prashasta Gujrati, Zhiling Lan, Xian-he Sun; Fault-driven re-scheduling for improving system level fault resilience, In: *Proceedings of the IEEE International Conference on Parallel Processing*, 2007.

[LZSSMG05] Yinglung Liang, Yanyong Zhang, Anand Sivasubramaniam, Ramendra K. Sahoo, Jose Moreira, Manish Gupta; Filtering Failure Logs for a BlueGene/L Prototype, In: *Proceedings of the Intl. Conf. on Dependable Systems and Networks (DSN)*, 476485, 2005.

[NaVo10] Meiyappan Nagappan, Mladen A. Vouk; Abstracting log lines to log event types for mining software system logs, In: *Mining Software Repositories (MSR)*, 2010 7th IEEE Working Conference, 2010.

[Sa05] Salfner, Felix; Predicting failures with hidden Markov models, In: *Proceedings of 5th European Dependable Computing Conference (EDCC-5)*, 41-46, 2005.

[Sa06] F. Salfner; *Modeling Event-driven Time Series with Generalized Hidden Semi-Markov Models*, Technical Report 208, Department of Computer Science, Humboldt University, Berlin, Germany, 2006.

[Sa08] Felix Salfner; *Event-based Failure Prediction: An Extended Hidden Markov Model Approach*, 2008.

[SSSZ04] R. K. Sahoo, A. Sivasubramaniam, M. S. Squillante, Y. Zhang; Failure data analysis of a large-scale heterogeneous server environment, In: *Proc. of IEEE Conf. on Dependable Systems and Networks (DSN)*, 2004.

[SaTs08] Felix Salfner, Steffen Tschirpke; Error Log Processing for Accurate Failure Prediction, In: *Proceedings of the First USENIX Workshop on the Analysis of System Logs*, 2008.

[So11] Stephen Sorkin; Splunk Technical Paper: *Large-Scale, Unstructured Data Retrieval and Analysis Using Splunk*, 2011.

[TsSi83] M. M. Tsao, Daniel P. Siewiorek; Trend analysis on system error files, In: *Proc. 13th International Symposium on Fault-Tolerant Computing*, 116119, Milano, Italy, 1983.

[ZLPG09] Ziming Zheng, Zhiling Lan, Byung H. Park, Al Geist; System log preprocessing to improve failure prediction, In: *International Conference on Dependable Systems and Networks*, 572-577, 2009.