stefan PRIEBSCH

PHP MIGRIEREN

KONZEPTE UND LÖSUNGEN ZUR MIGRATION VON PHP-ANWENDUNGEN UND -UMGEBUNGEN

HANSER

Inhalt

2

2 Strategien

Ob du eilst oder langsam gehst,
der Weg bleibt immer der gleiche.

(Chinesisches Sprichwort)

In der Informationstechnologie gibt es keine endgültigen Lösungen. Ebenso wie die reale Welt ändert sich auch die Welt der Informationstechnologie stetig und schnell. Software wird erweitert, weil neue Features benötigt werden. Software wird optimiert, weil Ergebnisse schneller berechnet oder mehr Benutzer gleichzeitig bedient werden sollen. Es werden Fehler oder Sicherheitslücken entdeckt, also muss Software geändert werden.

Auch die Hardware ändert sich stetig. Nach Moores Gesetz [Droess 2005] verdoppelt sich ungefähr alle 18 Monate die Anzahl der Transistoren in einem Mikroprozessor[1]. Spätestens alle paar Jahre ändern sich Bauform und Schnittstellen von Hauptplatinen, Speichern oder Festplatten. Die Anwender erwarten selbstverständlich, dass Software nicht nur auf der neuesten verfügbaren Hardware, sondern auch auf älteren und bereits in die Jahre gekommenen Systemen läuft.

Für Anwendungsentwickler ist es ist nicht immer einfach, mit dieser Entwicklung Schritt zu halten. Dank der Arbeit von Betriebssystemherstellern und den Treiberprogrammierern bleibt ein Großteil der Hardware-Änderungen vor den Anwendungsentwicklern verborgen. Dafür müssen sie jedoch stärker auf das sich stetig ändernde Software-Umfeld Rücksicht nehmen. Je stärker eine Anwendung mit dem Betriebssystem verzahnt ist, desto schwieri-

---

1 Moores Gesetz wird oft falsch zitiert. Nicht die Rechenleistung der Prozessoren verdoppelt sich alle 18 Monate, sondern die Anzahl der Transistoren. Natürlich bedeutet dies auch eine Steigerung der Rechenleistung, aber eben nicht unbedingt im gleichen Verhältnis.

ger ist es, bei gleichzeitiger Rückwärtskompatibilität zu älteren Versionen auch eine neue Betriebssystemversionen zu unterstützen.

Besonders schwer haben es die Systemadministratoren, die meist ein knappes Budget zur Verfügung haben, aber die Systeme immer wieder an ein sich schnell änderndes geschäftliches Umfeld anpassen müssen. Dabei müssen sie nach außen hin Kontinuität bewahren, da die Anwender ihre Gewohnheiten nur ungern ändern.

Die Migration von Software und Systemen spielt sich an der Schnittstelle zwischen Entwicklern und Administratoren ab. Um dabei erfolgreich zu sein, muss das zu migrierende System und dessen Umfeld ganzheitlich betrachtet werden. Auch die Anwender sollte man bei einer Migration niemals aus dem Blickfeld verlieren. Schließlich entscheiden sie, ob ein System benutzt wird oder nicht.

In diesem Kapitel beschäftigen wir uns mit verschiedenen grundlegenden Strategien zur Migration. Hierbei spielen Denkzwänge der Systemverwaltung ebenso eine Rolle wie die Denkzwänge der Softwareentwicklung. Aber auch die Anwender und das Management müssen sich trotz zu wenig Zeit und knapper Budgets darüber im Klaren sein, dass in der Informationstechnologie keine Lösung endgültig ist. Migrationen sind daher ein inhärenter Bestandteil des IT-Lebens.

Die Frage ist nicht, ob jemals wieder eine Migration notwendig sein wird, sondern wann die nächste Migration durchgeführt werden muss.

2.1 Never touch a running system

Jeder, der schon einmal ein System verwaltet hat, und sei es nur der Rechner am eigenen Arbeitsplatz, kennt das Problem: Bevor man sich in den Feierabend, ins Wochenende oder gar in den wohlverdienten Urlaub verabschiedet, soll nur noch schnell ein kleines Software-Update installiert, die Konfiguration des Rechners geändert oder ein neues Programm installiert werden.

Selbst wenn danach auf den ersten Blick die Welt noch in Ordnung scheint, ist damit nicht gesagt, dass nach dem nächsten Neustart des Systems noch alles wie erwartet funktioniert. Mitunter genügt eine Systemeinstellung, die nach einem Neustart nicht mehr wirksam ist oder erst durch einen Neustart wirksam wird, um Ihnen einige Stunden Stress und mühsame Fehlersuche zu bescheren.

Natürlich ist es reiner Aberglaube, aber manchmal könnte man den Eindruck gewinnen, der Rechner spürt es, wenn man es eilig hat oder kurz vor einem wichtigen Termin steht, und macht dann mit Vorliebe Ärger. Eine gern zitierte goldene Regel – wenn nicht *die* goldene Regel – der Informationstechnologie lautet daher: *Never touch a running system*.

Der Ursprung dieser Weisheit ist nicht in bekannt. Ich könnte mir allerdings durchaus vorstellen, dass sie sich ursprünglich auf mechanische Geräte bezog und man einfach verhindern wollte, dass jemand einen Finger verliert. In der Informationstechnologie gibt es – vielleicht abgesehen von Gehäuselüftern – nur wenige direkte Risiken für Finger, aller-

dings riskiert man hier oft den sprichwörtlichen Kopf und Kragen, wenn man ein unternehmenskritisches, laufendes System „anfasst“ und daraus Produktionsausfall resultiert.

2.1.1 Systemumgebung

Mit *Running System* sind in erster Linie laufende Produktivsysteme gemeint. Sofern ein System nur zu normalen Bürozeiten verfügbar sein muss, ist es einfach, die notwendigen Wartungen abends oder am Wochenende durchzuführen, wenn das System nicht benötigt wird. Wichtig ist, dass man in diesem Fall einen genügend großen Zeitpuffer einplant, um das System bei eventuellen Problemen wieder auf Vordermann zu bringen, bevor es produktiv benötigt wird.

In der heutigen Internet-Zeit müssen viele Anwendungen 24 Stunden am Tag und sieben Tage pro Woche verfügbar sein. Zeiten der Nichtverfügbarkeit müssen je nach Einsatzgebiet des Systems lange vorher angekündigt werden, was eine langfristige Planung der Wartungstätigkeiten erfordert. Um die geplanten Ausfallzeiten möglichst kurz zu halten, ist man gut beraten, alle Maßnahmen zuerst auf möglichst identischen Testsystemen zu überprüfen, bevor man eine Änderung am Produktivsystem vornimmt.

Leider sind in der Realität die entsprechenden Testsysteme nicht immer vorhanden, und bei entsprechender Komplexität des Produktivsystems werden diese schnell zu einem Kostenfaktor. Man kann zwar heute durch Virtualisierung von Systemen gerade bei Testsystemen erhebliche Hardwarekosten sparen, aber je mehr Unterschiede es zwischen dem Testsystem und dem Produktivsystem gibt, desto wahrscheinlicher ist es, dass auf dem Produktivsystem neue oder andere Probleme auftauchen[2].

Je älter Systeme werden, desto länger ist ihre Historie von installierter Software, aktualisierter Software, wieder entfernter Software und geänderter Konfiguration. Leider wird diese Historie in den seltensten Fällen wirklich sauber dokumentiert, sodass man meist nicht mehr mit Sicherheit sagen kann, mit welchen Installations- und Konfigurationsschritten man von der Grundinstallation zum aktuellen Systemzustand kommt. In Verbindung mit automatischen Updates von Programmen oder Systemkomponenten ist es natürlich besonders schwierig, den Überblick über den Versionsstand eines Systems zu behalten.

Je weniger ein solches System dokumentiert ist, desto besser ist man beraten, keine unnötigen, scheinbar harmlosen Änderungen daran vorzunehmen. Wenn nach einer solchen Änderung bestimmte Programme nicht mehr funktionieren und es nicht gelingt, das System wieder in einen Zustand zu bringen, in dem es wie erwartet funktioniert, steht man vor der Frage, wie man ein Ersatzsystem installieren und konfigurieren muss.

---

[2] In letzter Zeit gibt es den Trend, auch Produktivsysteme zu virtualisieren. Das macht es beispielsweise besonders einfach, eine wirklich identische Kopie eines Produktivsystems als Testsystem zu erstellen. Die virtuellen Maschinen haben eine einheitliche Hardware, die vom Wirtssystem weitgehend unabhängig ist. Dadurch sind die virtuellen Maschinen ohne Änderung auf unterschiedlichen physischen Systemen einsetzbar.

Sofern das ursprüngliche System – sozusagen als Vorlage – noch vorhanden und zugreifbar ist, ist es relativ einfach, im direkten Vergleich Alt gegen Neu ein identisches oder zumindest hinreichend ähnliches System zu installieren. Wenn Sie aber keine Ersatzhardware haben und deshalb das alte System tatsächlich neu installieren müssen, ist der Ausgang der Rettungsaktion ungewiss, zumal Sie sich jeglicher Rückfallposition berauben, sobald Sie die alte Installation überschreiben.

In jedem Fall müssen Sie ein Backup aller relevanten Daten und Einstellungen erstellen. Bei den heutigen Datenmengen kann es schon einige Stunden dauern, bis die relevanten Daten auf einen anderen Rechner übertragen sind. Am besten ist es natürlich, die originale Festplatte unversehrt aufzuheben und die neue Installation auf einer neuen Festplatte durchzuführen. Das funktioniert allerdings nur, wenn Sie direkten Zugriff auf den Rechner haben, und ist für Mietserver, die in Rechenzentren stehen, keine Option.

Never touch a running system wird oft als Entschuldigung dafür missbraucht, an einem System notwendige Updates und Aktualisierungen zu verschleppen. Es ist durchaus sinnvoll, an einem Produktivsystem nur Veränderungen vorzunehmen, die einen klaren Nutzen haben. Führen Sie dabei zuvor möglichst umfassende Tests auf einem anderen System durch, und sorgen Sie in jedem Fall dafür, dass Ihnen immer eine Rückfallposition bleibt, mit der Sie schnell und unkompliziert zu einem funktionierenden System zurückkehren können, falls tatsächlich etwas schiefgeht.

Sobald *Never touch a running system* zum Zwang wird, weil ein System nicht ausreichend dokumentiert ist, man nicht mehr weiß, was geschieht, wenn man am System Änderungen oder Aktualisierungen vornimmt, wird es sehr gefährlich, einfach weiter abzuwarten. Schon das nächste auftretende Problem kann durch eine ungünstige Verkettung von Abhängigkeiten Ihr gesamtes Systemkonzept wie ein Kartenhaus zusammenstürzen lassen, wie das folgende Beispiel zeigt.

Eines schönen Tages möchten Sie hinter dem Server sauber machen und beschließen, dabei gleich die Kabel zu ordnen. Sie fahren also den Server herunter, säubern die Ecke und schaffen dort Ordnung. Nach getaner Arbeit möchten Sie Ihren Server neu starten. Dieser verweigert allerdings den Dienst, da sich die Festplatte aus dem Leben verabschiedet hat[3].

Sie müssen also eine neue Festplatte besorgen. Leider stellen Sie fest, dass Sie kurzfristig keine Festplatte mit der verwendeten Schnittstelle bekommen können. Leider unterstützt das Motherboard Ihres Servers die aktuellen Festplatten nicht mehr. Da Sie nicht darauf warten können, bis die Ersatzfestplatte geliefert wird, müssen Sie kurzfristig auf eine andere Server-Hardware ausweichen.

Leider lässt sich die vollständige Sicherung ihres alten Servers nicht so ohne Weiteres auf dem neuen Server installieren, da dort wegen der geänderten Hardware völlig andere Treiber nötig sind. Da Sie kein instabiles System riskieren wollen, entscheiden Sie sich dage-

[3] Tatsächlich fallen Festplatten eher selten im laufenden Betrieb aus, sondern eben beim Hochfahren eines Systems. Das ist ein Grund, warum Server möglichst selten heruntergefahren werden und dort die Stromsparmaßnahmen für die Festplatten abgeschaltet sein sollten.

gen, das vorliegende Backup einzuspielen, sondern beschließen, den neuen Server von Grund auf neu zu installieren.

Nach zwei bis drei vergeblichen Installationsversuchen müssen Sie einsehen, dass das etwas in die Tage gekommene Betriebssystem Ihres alten Servers auf der neuen Hardware nicht mehr installierbar ist. Selbst wenn Sie die Installation abschließen könnten, wäre es fraglich, ob das System jemals stabil läuft.

Sie sind also gezwungen, eine neue Betriebssystemversion einzusetzen. Diese lässt sich auf dem neuen Server problemlos installieren, sodass Sie neue Hoffnung schöpfen. Kurz darauf stellen Sie allerdings fest, dass nicht alle benötigten Softwarekomponenten, die Ihre Anwendung als Infrastruktur benötigt, unter dem neuen Betriebssystem laufen.

Leider können Sie nicht einfach die neueste Version bestimmter Softwarekomponenten mit einer älteren Version anderer Komponenten kombinieren, da sich zwischen den Versionen die internen Schnittstellen geändert haben. Sie müssen also einheitlich eine relativ aktuelle Version aller benötigten Softwarekomponenten installieren.

Nun kommt es, wie es kommen musste: Ihre Anwendung läuft in der neuen Umgebung nicht oder zumindest nicht fehlerfrei. Sie haben schon Stunden damit verbracht, ein funktionierendes System zu installieren, und müssen nun noch auf unbestimmte Zeit versuchen, Ihre Anwendung in der neuen Umgebung zum Laufen zu bekommen. Noch schlimmer ist es, wenn die Fehler nicht so offensichtlich sind, dass sie sofort entdeckt werden. In diesem Fall kann Ihnen eine sehr unruhige Zeit mit ungewissem Ausgang bevorstehen.

Eine solche Situation ist für alle Beteiligten wirklich unangenehm. Letztlich sind Sie gezwungen, alle Versäumnisse der Vergangenheit im Rahmen einer großen Feuerwehraktion nachzuholen. Dies ist das Damoklesschwert, unter dem Sie stehen, wenn Sie zu lange nach der *Never touch a running system*-Philophie leben.

2.1.2 Programmcode

Oftmals wird die Regel *Never touch a running system* auch als Rechtfertigung dafür verwendet, bestehenden Programmcode nicht zu ändern. Diese Sichtweise hat zunächst einmal durchaus ihre Berechtigung, denn jede auch noch so unbedeutend erscheinende Änderung am Quellcode kann dazu führen, dass ein Programm nicht mehr funktioniert beziehungsweise nicht mehr wie erwartet funktioniert.

Es ist leider viel zu einfach, bei einer kleinen Änderung am Quellcode einen Syntaxfehler in das Programm einzubauen. In übersetzten Sprachen deckt der Compiler solche Fehler schon frühzeitig auf, da der Quellcode nicht mehr übersetzt werden kann. In PHP allerdings gibt es keinen Compiler, daher bleiben diese Syntaxfehler zunächst unentdeckt und treten erst dann zutage, wenn die fehlerhafte Datei auch tatsächlich geladen und ausgeführt wird.

Eine gute integrierte Entwicklungsumgebung zeigt Ihnen direkt während des Editierens an, welche Syntaxfehler eine PHP-Datei enthält. Aber auch ohne IDE sollten Sie versuchen, wenigstens solche offensichtlichen Fehler möglichst frühzeitig zu finden. Führen Sie daher

Register

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z