2 Grundlegende Optimierungsmöglichkeiten

Bevor die eigentliche Anwendung auf ihre Performance hin optimiert werden kann, muss zuerst die aktuelle Leistungsfähigkeit bzw. Performance der Anwendung ermittelt werden. Dieser Vorgang ist wichtig, da dadurch Schwachstellen im System aufgedeckt werden können. Zudem ist eine kontinuierliche Liveüberwachung von Vorteil, um schneller auf eventuell bevorstehende Performance-Schwächen aufmerksam zu werden und um frühzeitig mögliche Ausfälle zu erkennen.

Sollte der Ausfall eines oder mehrerer Systeme eintreten, ist eine Notfallstrategie unabdingbar: Hinzuschalten weiterer Ressourcen oder das Abschalten überlasteter Bereiche der Anwendung können Auswege aus solch einer Situation darstellen.

Nichtsdestotrotz helfen diverse Überwachungs- und Ausfallszenarien nur bedingt, wenn die Systeme unzureichend entwickelt und optimiert sind. Die Entwicklung eines hoch optimierten Systems beginnt bereits bei der untersten Ebene: der Hardware und dem Betriebssystem, das darauf läuft. Moderne Hardware ist in der Regel kompakter, verbraucht weniger Strom und ist performanter als die Vorgängerversionen. Aktuelle Betriebssystemversionen beinhalten nicht nur systemkritische Fehlerbehebungen, sondern in der Regel auch performancekritische Optimierungen. Zudem spielt das Softwaredesign ebenfalls eine wichtige Rolle. Hierunter zählt nicht nur die Architektur der Anwendung, sondern auch die Datenhaltung und die Benutzerführung (Usability) der Software. Zusätzliche Caching-Ebenen oder Queuing-Services können der Anwendung einen extra Performance-Schub geben.

2.1 Ermittlung der Performance

Systemoptimierungen sollten erst durchgeführt werden, wenn die Schwachstellen und Flaschenhälse der Anwendung bekannt sind. In Softwareprodukten kann dies bereits durch das Ermitteln der Laufzeit von Methoden durch Profiling erreicht werden. Hierfür gibt es sowohl für den Client

(Browser) als auch für den Server diverse Tools auf dem Markt. Neben dem Messen der Performance können weiterhin Lasttests durchgeführt werden, um die Stabilität des Systems bei hoher Last zu untersuchen. In Kapitel 3 finden sich dazu mehr Informationen.

Die Performance der Software ist nur ein Teil der Gesamtperformance einer Webanwendung: Netzwerk-, Server- oder Festplattenauslastung können vor allem bei *»Shared Web Hosting«*-Angeboten[1] unberechenbar sein und die Performance der eigenen Anwendung unterschiedlich stark beeinflussen. Um in einem solchen Szenario rechtzeitig auf bevorstehende Lastspitzen reagieren zu können, muss das System im Livebetrieb überwacht werden (siehe Abschnitt 3.4.5). Die Überwachungstools können bei bestimmten Ereignissen (Festplattenausfall, Netzwerküberlastung, hohe CPU-Last etc.) entsprechende Maßnahmen automatisiert ausführen (siehe Abschnitt 2.2.1) oder den Administrator z.B. über eine E-Mail informieren.

Außer der automatisierten Liveüberwachung können auch gezielt Komponenten auf ihre Performance hin untersucht werden. Bestimmte Teile der Anwendung werden hierbei in sogenannten »Performance-Tests« oder »Lasttests« auf ihre Leistungsfähigkeit unter definierten Testbedingungen oder extremen Überlastszenarien hin untersucht. Überwachungspunkte an bestimmten Komponenten im System, beispielsweise per Logging oder Monitoring, geben hierbei Auskunft über Auslastung, aufgetretene Fehler und mögliche Bottlenecks. Hierdurch können Schwachstellen frühzeitig erkannt und Gegenmaßnahmen eingeleitet werden (siehe dazu Kapitel 3).

2.2 Planung von Überlast- und Ausfallszenarien

2.2.1 Graceful Degradation

Eine mögliche Maßnahme, um auf Lastspitzen zu reagieren, ist die gezielte Abschaltung von Diensten, auch *»Graceful Degradation«* genannt. Hierbei wird zwischen der Abschaltung einzelner Features der Anwendung, der Umleitung von Systemressourcen von anderen Anwendungen oder der Limitierung der Zugriffe auf die Anwendung unterschieden.

Kann die betroffene Anwendung in mehrere voneinander unabhängige Features aufgetrennt werden, besteht der optimale Weg darin, bei Lastspitzen einzelne davon abzuschalten. Dies können z.B. die für das Geschäftsmodell weniger kritischen Features sein. Die Reihenfolge der Abschaltung wird durch die Priorität der einzelnen Funktionen bestimmt. Ein gutes Beispiel für die Abschaltung von Diensten ist Facebook, dessen Monitoring-Anwendung Lastspitzen auf einzelne Features der Anwendung erkennt und eher

[1] Bei *»Shared Web Hosting«*-Angeboten teilen sich mehrere Webseiten dieselben Serverressourcen, um dadurch eine effizientere Auslastung dieser zu gewährleisten.

unkritische Funktionen abschaltet [Haa11]. Da das Hauptgeschäftsmodell von Facebook der Umsatz durch die eingeblendete Werbung ist und somit von der Anzahl der Benutzer auf der Plattform abhängt, könnte die niedrigste Priorisierung z.B. auf der Chat-Funktion liegen. Bei einer Lastspitze könnte diese vorübergehend abgeschaltet werden, um die Ressourcen für die Basisfunktionalitäten wie z.B. Statusmeldungen zu verwenden.

Die Erkennung von Lastspitzen und die Reaktion auf diese bedarf viel Planung im Voraus und kann auf verschiedene Weisen umgesetzt werden. Ziel ist es, ein sich selbst überwachendes System zu entwickeln, das eigenständig auf Ereignisse und Lastspitzen reagieren kann, um die eigene Funktionsfähigkeit zu gewährleisten. Der Aufwand, um solch ein System zu entwickeln, ist immens und nicht unbedingt für alle Anwendungen vonnöten. In vielen Szenarien reicht es daher, sich über folgende Punkte Gedanken zu machen, um im Falle einer Lastspitze entsprechend reagieren zu können.

- Welche Features existieren in der Anwendung und welche können temporär abgeschaltet werden?
- Wie viele Ressourcen werden bei der Abschaltung der Features frei?
- Wie ist die Priorisierung der Features entsprechend dem Geschäftsmodell?

Zudem kann der Einsatz von Load Balancer oder virtuellen IP-Adressen für diverse Anwendungsfälle vorteilhaft sein. Load Balancer sorgen für eine Aufteilung der ankommenden Anfragen auf mehrere Server, um dadurch die Last von einzelnen Servern auf mehrere zu verteilen. Da Load Balancer eine kritische Komponente darstellen, über die der Großteil der Anfragen geleitet wird (engl. »single point of failure«), müssen diese Komponenten sehr leistungsstark sein und ggf. durch einen zweiten Load Balancer im »Hot Standby«[2] abgesichert werden. Virtuelle IP-Adressen werden zum Beispiel bei Komponenten im Hot Standby verwendet, um zeitnah nach einem Ausfall einer Komponente die IP-Adresse an die zweite Komponente zu übertragen. Dadurch kann die zweite Komponente automatisiert ohne großen Konfigurationsaufwand die Arbeit der ersten übernehmen.

Neben der Abschaltung von Features innerhalb der Anwendung können auch Ressourcen anderer Anwendungen abgezogen und für die sich unter Last befindende Anwendung verwendet werden. Kann auf diese Ressourcen nicht zurückgegriffen werden, gibt es noch die Möglichkeit, kurzfristig Systemressourcen, zum Beispiel bei Cloud-Diensten, zu mieten und hinzuzuschalten.

[2]Beim Hot Standby wird eine Komponente redundant vorgehalten oder betrieben, sodass diese beim Ausfall der primären Komponente sofort deren Arbeit übernehmen kann.

2.2.2 Cloud-Dienste

Die Nutzung von Systemressourcen aus der Cloud (z.B. Amazon EC2, Google App-Engine oder Microsoft Azure) ist eine gute und günstige Möglichkeit, um kurzzeitige Lastspitzen abzudecken. Selbst die Überbrückung einiger Wochen bei planbarer erhöhter Last (z.B. vor Weihnachten) kann damit kostengünstig realisiert werden. In vielen Fällen lohnt sich die Verwendung von Cloud-Diensten gegenüber dem Kauf neuer Hardware, die in der restlichen Zeit des Jahres nicht ausgelastet wird.

Von der Realisierung ganzer Anwendungen auf Cloud-Diensten ohne eigene Hardware wird jedoch abgeraten, da diese im Dauerbetrieb teuer sind und im Falle eines Ausfalls des Cloud-Anbieters die gesamte Anwendung nicht mehr verfügbar ist. Viele Start-ups setzen am Anfang auf Cloud-Dienste, da oftmals zu Beginn des Unternehmens aus Kostengründen keine eigene Hardware betrieben wird. Cloud-Anbieter sind in der Regel aber nur dann zu empfehlen, wenn kurzfristig Ressourcen benötigt werden. Bei einem Dauerbetrieb sind die Kosten für eigene Hardware meist geringer.

Der Datenschutz spielt bei der Überlegung des Einsatzes von Cloud-Diensten ebenfalls eine wichtige Rolle. Viele Anbieter sind ausländische Unternehmen, womit die Regelung des Datenschutzes dieser Länder für die gespeicherten Daten gilt. In den USA sind beispielsweise der »Digital Millenium Copyright Act« und »Patriot Act« zu nennen, die Behörden die Durchsuchung von gespeicherten Daten erlauben. Im Vergleich dazu hat Deutschland eines der restriktivsten Datenschutzgesetze, was bedeutet, dass gesammelte Daten auf Servern in Deutschland diesem Datenschutz unterliegen. Handelt es sich bei den Daten somit um sensitive Daten, sollte der Einsatz von Cloud-Diensten gut überlegt werden.

2.3 Client- und serverseitiges Softwaredesign

2.3.1 Verbesserung der Usability und Nutzerführung

Durch eine Verbesserung der Usability verkürzt sich der Weg (und die Zeit), die ein Besucher auf der Webanwendung benötigt, um sein Ziel zu erreichen. Jede Bewegung des Benutzers in der Anwendung erzeugt Anfragen und damit Last auf den Servern, dem Übertragungsweg und dem Client. Die Verkürzung des Weges erhöht zum einen die Zufriedenheit des Besuchers und reduziert zum anderen die Last auf die Anwendung. Lässt sich beispielsweise anhand von Logfile-Auswertungen feststellen, dass viele Nutzer von der Startseite über Unterseite A nach Unterseite B gelangen und erst dort die gewünschten Inhalte finden, wäre es möglicherweise sinnvoll, die Unterseite B direkt auf der Startseite zu verlinken. Dadurch muss der Nutzer die Unterseite A nicht mehr aufrufen. Dies entlastet die Server,

reduziert den Datentransfer und erspart den Nutzern überflüssige Klicks und Wartezeiten.

Welche Inhalte interessieren den Nutzer?

Als Beispiel für den Einfluss der Nutzerführung lässt sich eine Studie von Google anführen: Es sollte herausgefunden werden, ob nach einer Suchanfrage 10, 20, 25 oder 30 Suchergebnisse aufgelistet werden sollen. In einer daraufhin durchgeführten Nutzerbefragung gaben die Teilnehmer an, dass sie möglichst viele Suchergebnisse bevorzugen. Google erhöhte daher in einem A/B-Test[3] bei einigen Nutzern die Anzahl auf 30 Suchergebnisse pro Seite, um so die Unterschiede im Nutzerverhalten messen zu können. Wie sich herausstellte, gingen die Suchanfragen bei den Nutzern mit 30 Suchergebnissen um 20 % zurück, anstatt dass die Nutzer häufiger Ergebnisse anklickten. Google fand anschließend heraus, dass dieser Rückgang mit der längeren Ladezeit der Ergebnisliste zusammenhing. Eine Seite mit 30 Suchergebnissen brauchte doppelt so lange, bis sie zum Browser übertragen und dargestellt wurde [Sha08].

Dieses Beispiel macht deutlich, dass nicht nur rein technische Optimierungen wichtig sind. Vielmehr sollte auch geprüft werden, ob der Nutzer alle übertragenen Inhalte überhaupt benötigt. Weiterhin zeigt dies, dass den Wünschen von Kunden nicht immer blind gefolgt werden muss. Das Testen von Änderungen ist daher immer empfehlenswert.

Ein häufig genanntes Beispiel ist auch die Suchseite von Google: Anstatt umfangreicher Portalinhalte, wie es beispielsweise Yahoo anbietet, findet sich auf der Suchseite lediglich ein Eingabefeld, der Suchbutton und dezente Menüleisten (siehe Abbildung 2.1). Dadurch ist die Seite sehr schnell geladen und der Nutzer findet sofort das, was er in dem Moment benötigt: das Eingabefeld für seine Suchanfrage. Da sich der Cursor bereits im Eingabefeld befindet, kann der Nutzer direkt mit der Eingabe seiner Suchbegriffe beginnen. Diese Reduzierung auf das Wesentliche gilt als eines der Erfolgskriterien von Google.

Arbeitsabläufe in der Webseite optimieren

Eine weitere Optimierung der Benutzerführung besteht darin, scheinbar selbstverständliche Arbeitsabläufe vor dem Benutzer versteckt im Hintergrund auszuführen. Bietet die Webanwendung zum Beispiel eine Uploadfunktion für Bilder an, können die Dateien, kurz nachdem sie vom Benut-

[3] Bei einem A/B-Test werden die Benutzer in zwei Gruppen unterteilt. Diesen Nutzern werden unterschiedliche Inhalte angezeigt, um deren Auswirkung bewerten zu können.

Abbildung 2.1: Startseite von Google und Yahoo im Vergleich

zer ausgewählt wurden, bereits im Hintergrund an den Server hochgeladen werden. Der Hochladevorgang kann dabei in einem für den Benutzer unsichtbaren IFrame oder per JavaScript durchgeführt werden. Dadurch ist der Benutzer in der Lage, im Vordergrund weitere Bilder auszuwählen und ggf. bereits hochgeladene Inhalte mit Metainformationen zu versehen. Der Benutzer wird dadurch bei seinem Arbeitsablauf nicht unterbrochen und kann zügig sein Ziel erreichen. Ein weiteres Beispiel sind Online-(Text-)Editoren. Hierbei können automatisch in regelmäßigen Abständen die Inhalte gespeichert werden. Dadurch kann der Benutzer, zum Beispiel nach einem ungewollten Absturz des Browsers, die Arbeit an dem Dokument mit einem relativ aktuellen Stand fortsetzen, ohne komplett von vorne beginnen zu müssen. Allgemein steigt durch die Optimierung solcher Arbeitsabläufe die Benutzerzufriedenheit, da der Ablauf insgesamt durchdachter und benutzerfreundlicher wirkt.

2.3.2 Clientseitige Optimierung

Optimierung der Datenübertragung

Die Optimierung der Datenübertragung kann durch die Reduzierung der zu übertragenden Daten erreicht werden. Je weniger Daten übertragen werden, desto geringer ist die Übertragungsdauer und desto schneller wird die Anwendung dargestellt. Außerdem kann die Datenkompression des Browsers und Webservers verwendet werden (siehe Abschnitt 7.4.1), um die zu übertragenden Daten zu minimieren. Dies kann durch eine Einstellung am Webserver erreicht werden und bedarf keiner Änderung der Anwendung. Die Geschwindigkeit bei der Übertragung geht jedoch zulasten der CPUs, da die Komprimierung der Daten Rechenzeit des Webservers beansprucht. Allerdings ist diese vernachlässigbar, da CPUs heutzutage spezielle Hardwareerweiterungen für die Komprimierung von Daten beinhalten.

Des Weiteren kann das Zusammenfassen ähnlicher Inhalte auf dem Übertragungsweg weitere wertvolle Zeit einsparen. Darunter zählt zum Beispiel das Zusammenfassen von JavaScript-, Stylesheet- oder Grafikdaten (siehe Abschnitt 7.4), wodurch weniger Verbindungen zum Server aufgebaut werden müssen. Solche automatisierten Funktionalitäten werden heute zum Teil schon als Erweiterungen für Webserver wie z.B. Apache angeboten.

Optimierung der Darstellung im Browser

Die Optimierung des Darstellungsprozesses im Browser kann für eine schnellere Verarbeitung und Darstellung der Webseite sorgen, wodurch sich die gefühlte Ladezeit verringern lässt. Die Optimierung besteht darin, dem Browser die Webseiteninhalte in einer optimalen Art und Weise auszuliefern, sodass der Parser und Interpreter der Darstellungs-Engine des Browsers einen geringen Arbeitsaufwand hat. Es genügt beispielsweise bereits die fixe Angabe der Größeninformation von Grafiken, um deren Darstellung im Webseitenkontext zu beschleunigen. Zudem kann die optimierte Einbindung von externen Inhalten, wie zum Beispiel Stylesheet- oder JavaScript-Dateien, diesen Vorgang ebenfalls stark beeinflussen (siehe Abschnitt 8.2).

2.3.3 Datenhaltung

Für die Speicherung der Daten ist eine optimale Strukturierung und Datenhaltung von Vorteil. Daten sollten in einer logisch sinnvollen Struktur vorliegen, um in nachfolgenden Prozessen eine schnelle Abarbeitung zu ermöglichen. Des Weiteren erleichtert eine optimale Datenstruktur die Arbeit der Entwickler.

Im Vorfeld sollte erörtert werden, welche Daten essenziell sind, welche Daten wirklich in weiteren Prozessen benötigt werden und welche Daten selten oder gar nicht weiter verarbeitet werden. Überflüssige Daten sollten frühestmöglich verworfen und nicht gespeichert werden, um wertvolle Ressourcen zu sparen.

Die eigentliche persistente Datenhaltung spielt ebenfalls eine wichtige Rolle. Auf dem Markt gibt es diverse Speicherlösungen, angefangen bei Dateisystemen über Datenbanken bis hin zu hochperformanten Arbeitsspeicher- oder Clusterspeicherlösungen. Im Vorfeld sollte bereits zum frühestmöglichen Zeitpunkt eine für das Problem optimale Speicherlösung gesucht werden.

Bei Optimierungen sollte evaluiert werden, ob ein Wechsel der Speicherlösung Performance-Vorteile bringt. Relationale Datenbanksysteme werden aus Performance-Gründen in großen Systemen immer häufiger durch schemalose Datenbanklösungen ersetzt. Weitere Informationen zur Datenhaltung in Datenbanken und derer Optimierung finden sich in Kapitel 4.

2.3.4 Caching

Mit dem Begriff Caching ist in der Informatik das Zwischenspeichern von Inhalten auf einem in der Regel schnellen Speichermedium gemeint. Das Konzept des Cachens hat das Ziel, häufig verwendete Inhalte schnell zur Verfügung zu stellen, um diese nicht ständig von der Datenquelle laden zu müssen. Caching kann jedoch auch weitere Zwecke wie Kostenreduktion, Bandbreitenreduktion, Hardwareauslastung und Geschwindigkeitsverbesserung erfüllen und ist damit ein wichtiger Bestandteil der Performance-Optimierung von Anwendungen. Die Erstellung eines Caching-Konzepts und die Bestimmung von sinnvollen Inhalten für den Cache ist ein wichtiger Schritt bei der Implementierung von Caching. Dieser kann durch einfache Fragestellungen und durch Profiling der Anwendung (siehe Abschnitt 3.5) unterstützt werden.

- Welche Daten werden in der Anwendung verwendet und woher kommen diese Daten?
- Welche Daten werden am häufigsten geladen?
- Welche Daten haben die längste Ladezeit?
- Sind die geladenen Daten immer gleich oder unterscheiden sich diese bei jeder Anfrage?
- Wie lange werden die Daten benötigt?

In Kapitel 5 finden sich detaillierte Informationen zu Caching, der Software sowie der Erstellung eines Caching-Konzeptes und der Identifizierung sinnvoller Elemente.

2.3.5 Queuing-Services

Asynchrone Verarbeitung wird auf der Seite des Clients immer häufiger durch Ajax-Anfragen realisiert, um Daten parallel oder nur dann, wenn sie benötigt werden, zu laden. Serverseitig resultiert jedoch häufig jede Anfrage direkt in einer Anfrage an Drittkomponenten, wie z.B. Datenbanken, Dateisysteme oder APIs. Auch wenn die Anfragen clientseitig asynchron ausgeführt werden, sind die Zugriffe auf die Komponenten direkt und können somit eine große Last auf den Systemen verursachen. Mit Queuing-Services können diese Zugriffe entkoppelt werden, um somit nur die maximale Last auf einer Komponente zuzulassen, ohne diese bei Lastspitzen zu überlasten. Detailliert werden Queuing-Services in Kapitel 6 vorgestellt.

2.4 Skalierung

Der Begriff der Skalierung bedeutet in der Informatik, ein System mit Ressourcen zu erweitern, sodass dieses mehr Last verarbeiten kann. Oft wird im Zusammenhang mit der Skalierung auch der Begriff der Skalierbarkeit verwendet. Eine Software bzw. ein System ist skalierbar, wenn dieses durch Zugabe von mehr Hardware mehr Benutzer verarbeiten kann und dabei die maximale Anzahl der Benutzer nur durch die aktuell eingesetzte Hardware begrenzt wird.

Die Herausforderung in der Softwareentwicklung liegt darin, Software und Systeme zu entwerfen, bei denen mit einer steigenden Anzahl an Benutzern ein linearer Anstieg des Ressourcenverbrauchs einhergeht. Unendliche lineare Skalierbarkeit ist allerdings in der Realität nur selten zu erreichen, da ein Großteil der eingesetzten Software und Systeme mit einer steigenden Anzahl an Benutzern einen im Vergleich dazu größeren Ressourcenverbrauch verursachen. Zudem kommt erschwerend hinzu, dass die eingesetzte Software (z.B. Datenbanken) oftmals nur bedingt skalierbar ist. Dies zeigt sich darin, dass mit weiterer Hardware keine weitere Skalierbarkeit erreicht werden kann. Dies bedeutet auch, dass sich ab einem gewissen Zeitpunkt durch den Einsatz von mehr Hardware kein Performance-Gewinn mehr erzielen lässt.

Bei der Skalierung wird zwischen zwei Arten unterschieden: vertikale (engl. »scale up«) und horizontale Skalierung (engl. »scale out«). Bei der vertikalen Skalierung werden die eingesetzten Server, die noch nicht die maximale Hardware-Ausstattungsstufe hinsichtlich freier Arbeitsspeicherslots,

freier Festplattenplätze und Anzahl an CPUs erreicht haben, ausgebaut. Schwache Server, die nicht mehr ausgebaut werden können, werden bei der vertikalen Skalierung durch neue stärkere Server ersetzt. Somit besteht bei Systemen, die nur vertikal skaliert werden können, eine Limitierung der maximalen Leistungsfähigkeit. Zudem bedeutet vertikale Skalierung, dass Ausfallsicherheit nur durch mehrere passive Server erreicht werden kann, die nur während eines Ausfalls zum Einsatz kommen und vorher ungenutzt bleiben.

Horizontale Skalierung bedeutet hingegen die Erweiterung der bestehenden Hardware um weitere Hardware. Dabei wird die Last auf die verfügbare Hardware verteilt, um somit einen parallelen Betrieb zu ermöglichen. Um dies zu erreichen, müssen die Webanwendung und die beteiligten Softwaresysteme, wie beispielsweise Datenbanken, auf mehrere Server verteilt werden. Dies ist jedoch nur mit Software möglich, die für solche Einsatzzwecke entwickelt und ggf. optimiert wurde.

Neben der Skalierbarkeit ermöglichen mehrere Server für ähnliche oder gleiche Einsatzzwecke eine erhöhte Ausfallsicherheit des gesamten Systems. Fällt ein Server aus, können andere Server im Verbund die Aufgabe des ausgefallenen Servers übernehmen und den Betrieb weiter gewährleisten. Im Vergleich zu vertikal skalierten Servern, die bei einem Ausfall (ohne Redundanzen) einen Ausfall des Gesamtsystems nach sich ziehen, sind die Folgen bei paralleler Nutzung weniger schwerwiegend.

Beide Verfahren sind in Abbildung 2.2 nochmals beispielhaft dargestellt.

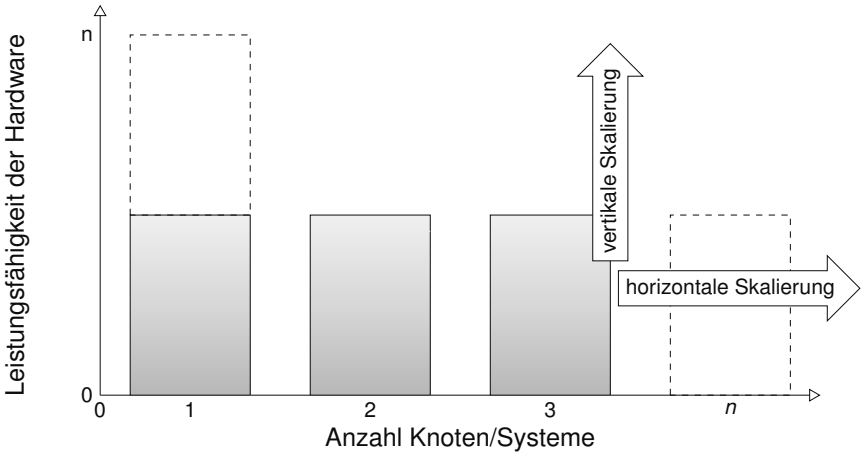

Abbildung 2.2: Vergleich horizontale und vertikale Skalierung