# Leistungsvermessung von I/O Systemen mit der MPI-IO-Schnittstelle

**Ralf Behnke, Daniel Versick and Djamshid Tavangarian**
([ralf.behnke | daniel.versick | djamshid.tavangarian]@informatik.uni-rostock.de)
Lehrstuhl für Rechnerarchitektur, Institut für Informatik,
Universität Rostock
Albert-Einstein-Str. 21, D-18059 Rostock, Deutschland

Da der Abstand der Leistung zwischen CPU und Sekundärspeicher ständig steigt, wird eine genaue und vergleichbare Leistungsanalyse des Sekundärspeicherzugriffs mittels I/O-Benchmarks immer wichtiger. Insbesondere in Hochleistungsrechensystemen müssen die immer leistungsfähigeren Prozessoren mit immer größeren Datenmengen versorgt werden, um eine optimale Ausnutzung der Ressourcen zu ermöglichen. Der Mangel derzeitiger I/O-Benchmarking-Systeme, relevante Messergebnisse zu liefern, soll mittels einer I/O-Benchmark-Suite, die MPI-IO-basierte Applikationen vermisst, überwunden werden. Diese Sammlung von Werkzeugen, die im vorliegenden Beitrag vorgestellt wird, ermittelt die I/O-Leistung anhand vorher ermitteltem Applikations-I/O-Verhalten und ermittelt so verlässliche und für Benutzer relevante Daten. Anhand eines Beispiels wird gezeigt, dass der vorgestellte Ansatz praktisch nutzbar ist.

**Schlüsselwörter:** I/O-Leistung, Verteilte Systeme, I/O-Profiling, I/O-Benchmark

## 1 Einleitung

In modernen Hochleitungsrechnern steigt die Rechenleistung und damit die Menge der verarbeitbaren Daten kontinuierlich an. Dabei erhöht sich die Geschwindigkeit der Datenspeicher nicht in gleichem Maße wie die Prozessorleistung, so dass es immer schwieriger wird, die CPU mit genügend Daten zu versorgen, um eine optimale Ausnutzung ihrer Rechenleistung zu erreichen. Zahlreiche Konzepte wurden entwickelt, um die mit diesem "Memory Wall"-Problem[1] verbundenen Nachteile zu minimieren. Beispielsweise werden die Speichergrößen von Cache-Speichern, die eine schnelle Zwischenspeicherung von Daten ermöglichen, erhöht. Ebenso werden neue Cache-Hierarchie-Stufen in die Speicher-Hierarchie eingeführt[2]. Am unteren Leistungsende der Speicherhierarchie steht aber immer der Sekundärspeicher, der die Daten letztendlich zur Verfügung stellt. Optimierungen an dieser Stelle wirken sich demnach auf die gesamte Speicherhierarchie aus. Dieser Beitrag stellt ein neues Benchmarksystem vor, das die Vermessung der I/O-Leistung von Sekundärspeicher durchführt und so zur Erkennung von Engpässen und damit zur Optimierung des Sekundärspeicherzugriffs verwendet werden kann. Derartige Benchmarks zur Vermessung der Sekundärspeicherleistung sollen in diesem Beitrag mit dem Begriff I/O-Benchmark bezeichnet werden.[3]

In Hochleistungsrechensystemen werden zum Sekundärspeicherzugriff besondere Schnittstellen verwendet. Gebräuchlich ist beispielsweise die Verwendung von MPI-IO als Teil des Message Passing Interfaces (MPI-2 Spezifikation [4]) der weithin gebräuchlichen standardisierten Schnittstelle zum Nachrichtenversand in verteilten Applikationen. MPI-IO ermöglicht einen optimierten parallelen Zugriff auf das Dateisystem von Clustercomputern und ist wesentlich komplexer als die typischerweise auf UNIX-Systemen verwendete POSIX-Schnittstelle. Aufgrund der höheren Komplexität der MPI-IO-Schnittstelle konzentriert sich der vorliegende Beitrag auf diese I/O-Schnittstelle. Die vorgestellten Konzepte sind durch Vereinfachung leicht auch auf andere I/O-Schnittstellen übertragbar.

### 1.1 I/O-Benchmarks

Da der Zugriff auf Sekundärspeicher über viele Ebenen der Speicher-Hierarchie abläuft und deshalb zahlreiche Subsysteme moderner Rechnersysteme involviert sind, ist die Ermittlung von Leistungsparametern der Speicher von einer großen Anzahl von Parametern abhängig. Dies sind nicht nur Parameter, die von der Hardware vorgegeben werden (wie beispielsweise die Bandbreite der Verbindung zwischen zwei Komponenten unterschiedlicher Speicher-Hierarchie-Ebenen), es sind auch Parameter, die durch Softwarekomponenten wie dem Betriebssystem vorgegeben werden. Dateisysteme als Teil des Betriebssystems sind die Schnittstelle des Benutzers zum Sekundärspeicher. Deren Implementierung ist also ebenfalls entscheidend für die I/O-Leistung eines Rechensystems. Aufgrund dieser extrem hohen Komplexität des I/O-Systems ist eine formale Ermittlung von I/O-Leistung undenkbar. Ein gängiger Ansatz zur I/O-Leistungsermittlung ist die Verwen-

dung von I/O-Benchmarks, die wie klassische Applikationen den Sekundärspeicher verwenden und dabei eine Leistungsermittlung im laufenden Betrieb durchführen. I/O-Benchmarks können anhand des von ihnen verwendeten Lastverhaltens in drei Arten klassifiziert werden. Die meisten heute verbreiteten I/O-Benchmarks gehören der Gruppe der *Benchmarks mit definiertem Workload* an. Sie verwenden einen festen durch den Benchmark-Entwickler definierten Workload, der vom Benchmark-Anwender nur geringfügig angepasst werden kann. Benchmarks mit definiertem Workload haben den Vorteil der leichten Nutzbarkeit aber den Nachteil, dass die Benchmarkergebnisse häufig für den spezifischen Anwender unbrauchbar sind, da die durch ihn ausgeführten Applikationen einen völlig anderen Workload verwenden. Eine zweite Klasse von I/O-Benchmarks erlaubt eine freie Konfiguration des Workloads und damit eine Anpassung an die Bedürfnisse des Nutzers. Der Nutzer muss bei Verwendung dieser *Benchmarks mit konfiguierbarem Workload* aber umfangreiche Kenntnisse über den I/O-Workload der Applikation besitzen, deren I/O-Verhalten vermessen werden soll. Im Rahmen dieses Beitrags wird ein Benchmark-System vorgestellt, das eine extrem flexible Konfiguration ermöglicht, den Benutzer aber mittels verschiedener Werkzeuge in der Analyse des I/O-Verhaltens einer Applikation und der anschließenden Vermessung unterstützt. Dieser Benchmark geht über die vorher genannten beiden Klassen hinaus und wird deshalb *applikationsbasierter Benchmark* genannt.[5]

### 1.2 Stand der Forschung

Die meisten existierenden I/O-Benchmarks wie der weit verbreitete *Bonnie* und *Bonnie++* Benchmark [6] verwenden fest definierte Workloads. Nur wenige Parameter wie beispielsweise die Größe der insgesamt transferierten Daten sind änderbar. Aus diesem Grund sind die Ergebnisse i. A. nur von geringem Nutzen. Selbst wenn der Benchmark mit dem vom Entwickler vorgeschriebenen Workload gute Ergebnisse auf einem Rechensystem erzielt, kann die I/O-Performance bei Verwendung des I/O-Workloads der letztlich auf dem Zielsystem zur Ausführung gebrachten Applikation sehr schlecht sein. Abbildung 1 verdeutlicht dies an Beispielmessungen. Dort wird die Lese- und Schreibleistung bei verschiedenen Größen von I/O-Anforderungen und bei zwei unterschiedlichen Mengen an insgesamt übertragenen Daten miteinander verglichen. Die Messungen wurden auf einem Intel-Pentium-III-System mit einem GB Hauptspeicher mit Hilfe des IOzone-Benchmarks [7] durchgeführt. Erkennbar ist, dass der Lesedurchsatz im Falle von einem GB an insgesamt transferierten Daten deutlich besser ist als der Schreibdurchsatz. Bei insgesamt 2 GB an transferierten Daten dreht sich dieses Verhältnis allerdings um. Dann ist der Schreibdurchsatz fast doppelt so hoch wie der Lesedurchsatz. Leistungsaussagen, die bei einem Workload getroffen werden können, sind also keineswegs auch bei allen anderen Workloads gültig. In diesem Fall ist der starke Durchsatzunterschied beider Messungen auf das Cacheverhalten des Systems zurückzuführen. Ein GB an Daten verursachen nur wenig Verdrängung im Hauptspeicher des Systems, der als Sekundärspeichercache dient, während 2 GB Daten bereits zu ständigen Verdrängungen führen.

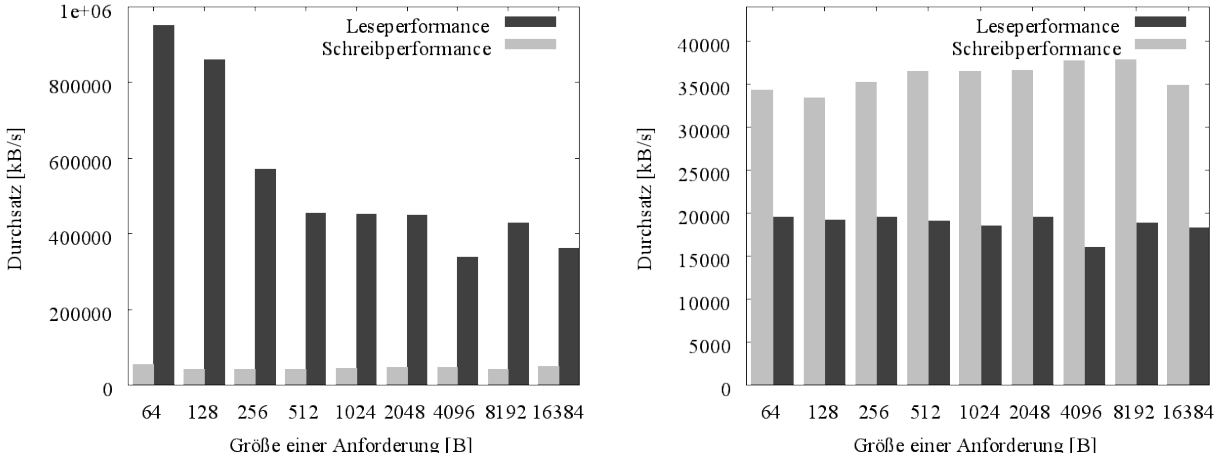
*IOzone* – der I/O-Benchmark, der für die gezeigten Beispielmessungen verwendet wurde, ist ein weithin verwendeter I/O-Benchmark. Er bietet die Möglichkeit komplexe Workloads, die auf dem I/O-Verhalten von Applikationen basieren, anhand zweier Konfigurationsdateien zu definieren. Natürlich muss der Anwender das Applikationsverhalten sehr genau kennen, wenn er diese Möglichkeit nutzen möchte. IOzone bietet keine Möglichkeiten, den Nutzer in der Erstellung der komplexen Konfigurationen zu unterstützen.[7]

Während die bisher vorgestellten I/O-Benchmarks ausschließlich POSIX-I/O verwenden, folgen nun zwei bekannte MPI-Benchmarks. Allerdings sind deren Konfigurationsmöglichkeiten und damit die vermessbaren I/O-Workloads nur sehr eingeschränkt. *NAS BTIO* als Teil der *NAS Parallel Benchmarks* verwendet einen sehr speziellen Workload, der berechnete Matrizen auf den Sekundärspeicher schreibt, reorganisiert und anschließend wieder einliest[8]. Die *Intel MPI Benchmarks (IMB)*, die vor dem Kauf der Firma Pallas durch Intel als *Pallas MPI Benchmarks (PMB)* bekannt waren, enthalten ebenfalls einen I/O-Benchmark, den *IMB-IO*[9]. Dieser testet zwar den größten Teil der MPI-IO-Zugriffsfunktionen, allerdings nicht in durch den Benutzer definierbaren Reihenfolgen, so dass auch hier der Gesamt-I/O-Workload sehr festgelegt und in diesem Fall aufgrund seines synthetischen Charakters für nahezu keine Applikation relevant ist.

Es zeigt sich also, dass die meisten der vorhandenen I/O-Benchmarks, insbesondere der MPI-IO-Benchmarks, praktisch zwar Ergebnisse liefern, diese aber für den Nutzer wenig relevant sind, da sie nicht dem I/O-Verhalten und damit auch nicht der I/O-Leistung relevanter Applikationen entsprechen. Dieser Beitrag stellt eine Benchmarksuite vor, die die adressierten Probleme beseitigt.

## 2 Benchmark-Architektur

Dieser Beitrag stellt eine neuartige Werkzeug-Suite vor, die den Nutzer in der akkuraten Messung der I/O-Leistung von MPI-IO-Programmen unterstützt. Wie Abbildung 2 zeigt, besteht diese Suite aus zwei Anteilen. Dies ist einerseits ein I/O Profiler, der das komplette I/O-Verhalten einer Applikation, auch wenn diese nicht im Quellcode zur Verfügung steht, bestimmt. Ergebnisse dieses Profilers sind die Basis




(a) bei 1 GB transferierten Daten

(b) bei 2 GB transferierten Daten

Fig. 1. Vergleich des I/O-Datendurchsatzes bei unterschiedlichen Zugriffsmustern

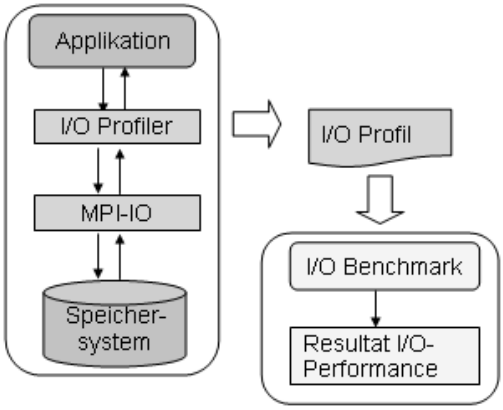



Fig. 2. Benchmark-Architektur

für Workload-Dateien, die als Eingabe für hochkonfigurierbare Benchmarks dienen können.

Andererseits wird in diesem Beitrag ein I/O-Benchmark präsentiert, der die Applikationsworkloads anhand der entstandenen I/O-Profile originalgetreu nachbilden kann und so genaue Daten über die I/O-Leistung von beliebigen Applikationen sammelt. Absatz 2.2 stellt die Implementierung dieses Benchmarks vor.

Absatz 3 zeigt erste Messungen, die bestätigen, dass die neue Benchmark-Suite eine gute Basis zum Leistungsvergleich unterschiedlicher Computersysteme in Hinblick auf spezifische Applikationen ist.

## 2.1 MPI-IO Profiler

Erster Teil der Werkzeugsammlung, die in diesem Beitrag vorgestellt wird, ist ein MPI-IO-Profiler, dessen Aufgabe die Protokollierung von MPI-IO-Aufrufen einer Applikation ist. Ziel dieses Profilers ist es, so viel Informationen wie möglich über die I/O-Aufrufe einer bestimmten Applikation zu sammeln, um so eine möglichst exakte Nachbildung des I/O-Verhaltens dieser Applikation zu erreichen.

Dabei ist es wichtig, dass Benutzer die Möglichkeit haben, I/O-Profile von Applikationen auch dann zu erstellen, wenn die Applikation nicht im Quellcode zur Verfügung steht. Deshalb wurde der Profiler als dynamische Bibliothek für Linux-Systeme konzipiert, die als Wrapper zwischen den Prozessen der verteilten Applikation und der darunterliegenden MPI-Bibliothek liegt. Diese so genannte Profiling-Bibliothek stellt der Applikation dazu die gleichen Funktionen zur Verfügung wie die MPI-Bibliothek, protokolliert aber zusätzlich jeden Aufruf der originalen MPI-Funktion.

Technisch wird die Wrapper-Funktionalität dadurch realisiert, dass vor dem Programmstart die Umgebungsvariable `LD_PRELOAD` verändert wird, die den dynamischen Linker veranlasst, die Profiler-Bibliothek, statt der Standard-MPI-Bibliothek gegen die Applikation zu linken. Dies macht es möglich, das I/O-Verhalten jeder Applikation zu protokollieren, die dynamisch gegen die MPI-Bibliothek gelinkt ist.

Der Vorgang der Protokollierung ist transparent für den Benchmarknutzer. Er startet die Applikation mit Hilfe eines neuen Startskriptes, das versucht, den Ort der MPI-Bibliothek zu finden und nach der Änderung der Umgebungsvariable `LD_PRELOAD` die entsprechenden Informationen an die Profiling-Bibliothek übergibt. Die Profiling-Bibliothek nutzt diese Informationen, um die originale MPI-Bibliothek zur Laufzeit zu laden und so die Originalfunktionalität des Message-Passing-Interfaces durch Funktionsaufrufe nutzen zu können.

Um das Applikationsverhalten so wenig wie möglich zu beeinflussen, kommunizieren die Profilingbibliotheken, die gegen die einzelnen Prozesse der Applikation gelinkt sind, während der eigentlichen Programmausführung nicht miteinander. Die einzelnen Profilerbibliotheken erzeugen ein I/O-Profil pro Prozess und schreiben dieses in einem CSV-angelehnten Format (Comma Separated Values) mit

Hilfe von POSIX-I/O-Funktionen. Nur so entstehen durch die Protokollierung keine zusätzlichen MPI-IO-Aufrufe, die das originale Applikationsverhalten beeinflussen könnten. Pro I/O-Anfrage werden die Zeit seit Programmstart vor und nach dem I/O-Aufruf, der Funktionsname, Rückgabewert und die übergebenen Parameter gespeichert. Die Zeiten sind insbesondere von Bedeutung, um die Reihenfolge der I/O-Aufrufe unterschiedlicher Prozesse in Hinblick auf die komplette Applikation nachvollziehen zu können. Ein komplettes Applikationsprofil wird erst am Ende des Applikationslaufes durch Aufruf der Funktion `MPI_Finalize` erzeugt. Der Wrapper dieser Funktion vereint die einzelnen Profile zu einer Datei, die Informationen über das komplette Applikations-I/O-Verhalten enthält.

Diese leicht portierbare und weitestgehend plattformunabhängige Implementierung des I/O-Profilers ermöglicht eine Nutzung der Software auf verschiedenen Plattformen mit unterschiedlichen MPI-Implementierungen ohne größere Änderungen.

### 2.2 Headstrong Benchmark

Zur Ermittlung der I/O-Performance eines Applikations-I/O-Profils wurde ein Benchmark entwickelt, der die eben eingeführten MPI-IO-Profile wiedergeben kann und die Zeit zur Wiedergabe der einzelnen I/O-Anfragen misst. Dieser so genannte *headstrong*-Benchmark ist Teil der PRIOmark Benchmark Suite [10], die an der Universität Rostock im Rahmen des IPACS-Projektes entwickelt wurde.[11]

Der *headstrong*-Benchmark nutzt POSIX-I/O zum Zugriff auf die Profilinformationen. Um eine adäquate Wiedergabe des I/O-Verhaltens der Originalapplikation zu ermöglichen, benötigt der *headstrong*-Benchmark während der Ausführung genau so viele Prozesse wie während der Applikationsaufzeichnung. Wird er mit mehr Prozessen gestartet, als während der Aufzeichnung zur Verfügung standen, nutzt er nur einen Teil der zur Verfügung stehenden Prozesse. Sollte er mit weniger Prozessen ausgeführt werden, ist eine adäquate Abbildung der Originalfunktionalität nicht mehr möglich. Der Benchmark wird dann mit einer Fehlermeldung beendet. Dieser Ansatz ist nicht nur für die adäquate Verhaltenswiedergabe wichtig. Er ermöglicht auch einen realistischen Vergleich der Leistung des I/O-Systems auf unterschiedlichen Architekturen.

Nach erfolgreicher Verteilung der Profildaten werden die Profilingbibliotheken der einzelnen Prozesse initialisiert. Insbesondere müssen dafür MPI-Kommunikatoren und MPI-Datentypen aufgebaut werden, um das Verhalten der Originalapplikation möglichst exakt nachbilden zu können.

Während der anschließenden Wiedergabe der Profile werden zahlreiche Performancedaten gesammelt und dem Nutzer präsentiert. Dieser kann an dieser Stelle zwischen fünf verschiedenen Granularitätsebenen wählen, je nachdem wieviele Leistungsdaten er benötigt bzw. möchte.

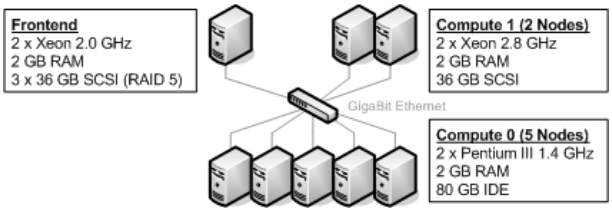



Fig. 3. Testumgebung

Der *headstrong*-Benchmark liest und schreibt genausoviele Daten wie die Originalapplikation. Lediglich der Inhalt der transferierten Daten, der zufällig erzeugt wird und die verwendeten Dateinamen differieren. Um Tests mit verschiedenen Dateisystemen in unterschiedlichen Pfaden des Verzeichnisbaums zu ermöglichen, erzeugt der Benchmark die Dateinamen während der Ausführung und erlaubt die Speicherung an benutzerdefinierten Pfaden. Zusätzlich führt der *headstrong*-Benchmark einen Vergleich mit Performancedaten durch, die während der Aufzeichnungszeit des I/O-Profiles mit dem Profiler erzeugt werden.

Bislang wurden die Protokollierung sämtlicher MPI-IO-Lese-, Schreib- und Seek-Funktionen (inklusive der non-kollektiven und asynchronen Versionen sowie alle Varianten davon) realisiert. Weniger wichtige MPI-Funktionen, die für eine exaktere Nachbildung des I/O-Verhaltens von Bedeutung sind, werden in zukünftigen Arbeiten ergänzt.

## 3 Ergebnisse

Dieser Absatz präsentiert Messergebnisse, die mittels des vorgestellten Ansatzes – der Verwendung von I/O-Profilen zur Simulation des Applikations-I/O-Verhaltens – ermittelt wurden. Dazu wurde das I/O-Profil einer Beispielapplikation erstellt, die MPI-IO verwendet und die I/O-Leistung der Applikation anschließend mit dem *headstrong*-Benchmark ermittelt. Um einen Vergleich der wirklichen Applikationsleistung mit der simulierten Leistung durchführen zu können, diente als Applikation ein herkömmlicher MPI-IO-Benchmark, der selbst seine I/O-Leistung ermitteln kann. Beide Messungen sollten im Vergleich ähnliche Werte zeigen, wenn der vorgestellte Ansatz funktioniert. Der in diesen Messungen verwendete Benchmark war das *strided*-Plugin [12] des PRIOmark Parallel I/O-Benchmarks.

Die I/O-Leistungsmessung wurde auf dem in Abbildung 3 abgebildeten PC-Cluster durchgeführt. Von den dargestellten Knoten wurden in den Messungen allerdings maximal 4 verwendet. Alle Knoten sind mit einem Linux Betriebssystem und der MPICH2-Implementierung von MPI ausgestattet. Außerdem sind die Dateisysteme NFS (Network Fi-

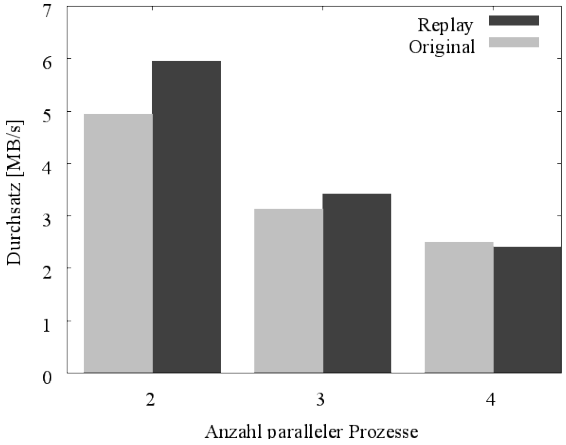



Fig. 4. Vergleich des I/O-Durchsatzes bei Ausführung des originalen *strided*-Benchmarks und bei Wiedergabe des entsprechenden I/O-Profils unter Verwendung des NFS-Dateisystems

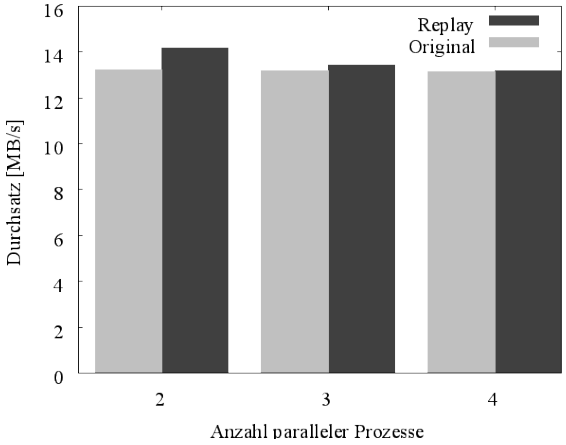



Fig. 5. Vergleich des I/O-Durchsatzes bei Ausführung des originalen *strided*-Benchmarks und bei Wiedergabe des entsprechenden I/O-Profils unter Verwendung des PVFS-Dateisystems

le System) und PVFS (Parallel Virtual File System) vorhanden, um einen Vergleich dieser Dateisysteme zu ermöglichen. Im Folgenden werden Messungen präsentiert, die mit Hilfe des *strided*-Benchmarks auf dem vorgestellten Cluster durchgeführt wurden und die beiden Dateisysteme NFS und PVFS vergleichen. Da der *strided*-Benchmark ein reiner MPI-IO Benchmark ist, sind die Ergebnisse dieses Benchmarks unbeeinflusst von anderen MPI-oder POSIX-Aufrufen. Mittels des vorgestellten MPI-IO-Profilers wurde eine Profildatei des I/O-Verhaltens des *strided*-Benchmarks erstellt, das als Basis für die Messungen diente. In diesen wurde das I/O-Verhalten des *strided*-Benchmarks anhand des I/O-Profils simuliert.

Abbildung 4 zeigt die Ergebnisse dieser Messungen bei Verwendung des NFS-Dateisystems. Die hellen Balken zeigen die Bandbreite pro Prozess, die durch den originalen *strided*-Benchmark bei I/O-Operationen erreicht werden konnten in Abhängigkeit von der Prozessanzahl. Die dunkleren Balken hingegen zeigen die Messergebnisse der Simulation des *strided*-Benchmarks durch den *headstrong*-Benchmark. Es ist deutlich erkennbar, dass die Ergebnisse sehr nah beeinander liegen. Lediglich bei zwei Prozessen ist die Abweichung der Bandbreite von ca. 1 MB/s relativ hoch. Weiterhin ist deutlich erkennbar, dass NFS, das auch bei Cluster-Computern noch immer sehr weit verbreitet ist, eine sehr schlechte Skalierung mit der Anzahl der Prozesse erreicht. Bei der Verwendung von 4 Prozessen ist die I/O-Bandbreite pro Prozess nur noch halb so groß wie bei der Verwendung von zwei Prozessen. Dies liegt an dem Flaschenhals, der beim Zugriff auf den NFS-Server auftritt. Da NFS nur einen NFS-Server verwendet, müssen folglich alle Prozesse über das Netzwerk auf diesen Server zugreifen[13].

Das Parallel Virtual Filesystem (PVFS) [14] ist dagegen auf den gleichzeitigen Zugriff mehrerer Prozesse optimiert. Es verteilt einzelne Dateien über mehrere I/O-Knoten. Der einzige Flaschenhals, der sich in einem PVFS ergeben kann, entsteht durch den Metadaten-Server, von dem es wiederum nur einen gibt. Dieser hält sämtliche Informationen über die Verteilung von Dateien über die Knoten, d. h. der Metadaten-Server muss bei jedem Öffnen einer Datei angefragt werden. Abbildung 5 zeigt die Ergebnisse der Messungen bei Verwendung eines PVFS-Dateisystems. Der verwendete Workload ist derselbe wie bei den in Abbildung 4 dargestellten NFS-Messungen. Deutlich ist die bessere Skalierung von PVFS mit der Anzahl der Prozesse erkennbar. Die I/O-Bandbreite beim Zugriff auf den Sekundärspeicher sinkt in diesem Beispiel nur unwesentlich bei steigender Prozessanzahl. PVFS auf dem verwendeten Cluster nutzt sämtliche Rechenknoten auch als I/O-Knoten, so dass bei 4 Prozessen auch noch eine maximale Parallelität des Zugriffes gewährleistet ist. Dies kann sich allerdings ändern sobald die Prozessanzahl die Anzahl der I/O-Knoten übersteigt. Bei den PVFS-Messungen ist die Genauigkeit des in diesem Beitrag vorgestellten I/O-Benchmarking-Ansatzes noch deutlicher erkennbar. Die Abweichungen sind mit ca. 6 % bei 2 Prozessen am größten.

Diese ersten Messungen zeigen, dass der beschriebende Ansatz des I/O-Benchmarkings beliebiger Applikationen anhand des aufgezeichneten I/O-Verhaltens funktioniert. Allerdings werden in Zukunft weitere Messungen durchgeführt, um die derzeit noch existierenden Abweichungen weiter zu minimieren und so den vorgestellten Ansatz zu verbessern.

# 4 Zusammenfassung

In diesem Beitrag wird das Problem vieler aktueller I/O-Benchmarks erörtert, dass diese nur einen statisch durch den Benchmarkentwickler definierten I/O-Workload verwenden. Es wird ein Ansatz vor-

geschlagen, einen I/O-Profiler zur Aufzeichnung von Applikations-I/O-Verhalten in Verbindung mit einem I/O-Benchmark zu verwenden, der dieses aufgezeichnete I/O-Verhalten möglichst realitätsnah wiedergibt und dabei Performancewerte sammelt. Dieser Ansatz erlaubt einen Vergleich verschiedener Architekturen bei der Nutzung spezieller Applikationen.

In dem Beitrag wird eine Implementierung eines derartigen Softwaresystems vorgestellt, das das MPI-IO-Interface verwendet und so verteilte Applikationen vermessen kann. Es werden Ergebnisse vorgestellt, die zeigen, dass die so ermittelten Performancewerte sehr genau den originalen Performancewerten entsprechen.

Das so entwickelte Benchmarksystem, das den sogenannten *applikationsbasierten Benchmarks* zugeordnet ist, ist ein möglicher Weg um die Vergleichbarkeit und Genauigkeit von I/O-Benchmark-Ergebnissen bei gleichzeitig geringem Messaufwand zu erhöhen.

# References


[1] McKee, Sally A.: Reflections on the memory wall. In: *CF '04: Proceedings of the 1st conference on Computing frontiers*. New York, NY, USA : ACM Press, 2004. – ISBN 1–58113–741–9, S. 162

[2] Krietemeyer, Michael ; Versick, Daniel ; Tavangarian, Djamshid: A Mathematical Model for the Transitional Region Between Cache Hierarchy Levels. In: Böhme, Thomas (Hrsg.) ; Rosillo, Victor M. L. (Hrsg.) ; Unger, Herwig (Hrsg.) ; Unger, Helena (Hrsg.): *Innovative Internet Community Systems*, Springer, 2004, S. 178–188

[3] Chen, Peter M. ; Patterson, David A.: A New Approach to I/O Performance Evaluation – Self-Scaling I/O Benchmarks, Predicted I/O Performance. In: *ACM Transactions on Computer Systems* 12 (1994), Nr. 4, S. 308–339

[4] Gropp, William ; Huss-Lederman, Steven ; Lumsdaine, Andrew ; Lusk, Ewing ; Nitzberg, Bill ; Saphir, William ; Snir, Marc: *MPI — The Complete Reference: Volume 2, the MPI-2 Extensions*. MIT Press `http://mitpress.mit.edu/book-home.tcl?isbn=0262571234`

[5] Krietemeyer, Michael ; Versick, Daniel ; Tavangarian, Djamshid: Workload-basierte Klassifikation von Benchmarks für lokale und verteilte I/O-Systeme. In: Bernd Wolfinger, Klaus H. (Hrsg.): *Leistungs-, Zuverlässigkeits- und Verlässlichkeitsbewertung von Kommunikationsnetzen und verteilten Systemen*, Universität Hamburg, Fachbereich Informatik, 2005, S. 119–127

[6] Bonnie Benchmark: *IOzone Filesystem Benchmark*. http://www.textuality.com/bonnie, 2005

[7] IOzone Filesystem Benchmark: *IOzone Filesystem Benchmark*. http://www.iozone.org/, 2005

[8] Wong, Parkson ; Wijngaart, Rob Van d.: NAS Parallel Benchmarks I/O Version 2.4. 2003. – Forschungsbericht

[9] GmbH, Intel: *Intel MPI Benchmarks - Users Guid and Methology Description*. Hermuehlheimer Str. 8a, D-50321 Brühl, Germany: Intel GmbH, 2006

[10] Krietemeyer, Michael ; Versick, Daniel ; Tavangarian, Djamshid: The PRIOmark – Parallel I/O Benchmark. In: *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks*, 2005, S. 595 – 600

[11] Falcone, Giovanni ; Kredel, Heinz ; Krietemeyer, Michael ; Merten, Dirk ; Merz, Matthias ; Pfreundt, Franz-Joseph ; Simmendinger, Christian ; Versick, Daniel: Integrated Performance Analysis of Computer Systems (IPACS). In: *Praxis der Informationsverarbeitung und Kommunikation* (2005), Nr. 3, S. 154–163

[12] Krietemeyer, Michael ; Kopp, Heiko ; Versick, Daniel ; Tavangarian, Djamshid: Environment for I/O Performance Measurement of Distributed and Local Secondary Storage. In: *Proceedings of the International Conference on Parallel Processing Workshops*, IEEE, 2005. – ISBN 0769523811, S. 501–508

[13] Pawlowski, Brian ; Juszczak, Chet ; Staubach, Peter ; Smith, Carl ; Lebel, Diane ; Hitz, Dave: NFS Version 3: Design and Implementation. In: *USENIX Summer*, 137-152

[14] Carns, Philip H. ; Ligon III, Walter B. ; Ross, Robert B. ; Thakur, Rajeev: PVFS: A Parallel File System for Linux Clusters. In: *Proceedings of the 4th Annual Linux Showcase and Conference*. USENIX Association, 317–327

[15] Falcone, G. ; Geiger, A. ; Kredel, H. ; Merten, D. ; Meuer, H. ; Meuer, M. ; Pfreundt, F. ; Kreuter, S. ; Reinig, D. ; Ristau, H. ; Simmendinger, C. ; Tavangarian, D. ; Versick, D. ; Merz, Matthias (Hrsg.) ; Krietemeyer, Michael (Hrsg.): *IPACS Benchmark - Integrated Performance Analysis of Computer Systems*. Logos Verlag, 2006. – 212 S. – ISBN 3832512535


### Danksagung


Diese Forschung wurde vom Bundesministerium für Bildung und Forschung unter der Fördernummer 01IRB03 im Rahmen des IPACS Projektes gesponsort. [15]