Projekt Parallelrechnerevaluation

Ausarbeitung

Johannes Pietrzyk

21. Februar 2012

Inhaltsverzeichnis

1 Übersicht

1.1 Ziel

Ziel dieser Ausarbeitung ist es, die Auswirkungen verschiedener Parameter des Benchmarks „HPC-LinPack“ auf den Energieverbrauch zu evaluieren.

1.2 Durchführung

Die Leistungsmessungen wurden mittels HPC-LinPack auf Clustern durchgeführt und die Ergebnisse (GFlops) auf starke Schwankungen untersucht. War es feststellbar, dass bei einem Parameter oder bestimmten Konstellationen erhebliche Abweichungen auftraten, so wurden diese Läufe fünfmal durchgeführt und das arithmetische Mittel über Leistung (GFlops) und Energieverbrauch (Watt beziehungsweise Joule) gebildet.

2 Der HPC-LinPack-Benchmark

HPC-LinPack ist ein speziell für Clusterrechner angepasster LinPack-Benchmark. LinPack löst ein lineares Gleichungssystem $Ax = b$, wobei

1. A : relle (m,n)-Matrix
2. x : reeller Vektor (m)
3. b : reeller Vektor (n)

mittels Gauß'schen Eliminationsverfahrens (Herstellung der Stufenform (obere Dreiecksmatrix) mittels elementarer Zeilenumformungen) und partieller Pivotisierung (Auswahl des betragsmäßig größten Elementes bezüglich der betrachteten Spalte).

- LR-Zerlegung: Schrittweise Berechnung der oberen Dreiecksmatrix R aus A und untere Dreiecksmatrix der Eliminationsfaktoren, sodass $A = L \cdot R$. Dient der Berechnung für beliebige rechte Seiten b.
- Spaltenpivotsuche: Das Pivotelement für den nächsten Eliminationsschritt wird mittels einer partiellen Suche gefunden, die für eine gegebene Spalte (deshalb partiell) das betragsgrößte Element findet und gegebenenfalls Zeilen vertauscht.
- Indexvektor: Statt tatsächlich Zeilen zu vertauschen, wird ein Indirektionsschritt in Form dieses Vektors eingeführt, der die Permutation der Indizes darstellt, die durch Vertauschen entstehen würden. Dieser Vektor wird dann für den Zugriff verwendet. Kann durch geschachtelte Arrays simuliert werden.

2.1 Eigenschaften

2.1.1 Aufteilung der Matrix

Die Rechenknoten werden als 2D-Gitter (rechteckig, beschrieben durch $P \cdot Q$) betrachtet. Die Matrix wird nun in quadratische Blöcke der Größe $nb \cdot nb$ aufgeteilt, die nach dem Verfahren der zweidimensionalen blockzyklischen Verteilung auf die Rechenknoten aufgeteilt werden, um möglichst gute Lastenverteilung zu erreichen.

- zweidimensionale zeilenzyklische Verteilung: Sei $1 \leq q \leq p, n = pq$ und wir betrachten Prozessor P_q, dann erhält dieser die Zeilen $q, q + p, q + 2p, ...$; dies ist o.g. Verteilung.
- zweidimensionale blockzyklische Verteilung: Hier werden ganze Teile der Matrix (rechteckige Blöcke) an die verschiedenen Rechenknoten verteilt. Dies wird durch einen Verteilungsvektor $(p1, b1), (p2, b2)$ beschrieben, wobei $p1 =$ Anzahl der Prozessoren entlang der Zeilen, $p2 =$ Anzahl der Prozessoren entlang der Spalten, $b1$ beziehungsweise $b2 =$ Blockgrößen der Zeilen beziehungsweise Spalten. In LinPack entspricht $(p1, p2) \rightsquigarrow P \cdot Q$ und $(b1, b2) \rightsquigarrow (nb, nb)$ (quadratische Blöcke). Die Gruppen sind disjunkt.
- Superblock: Dies ist wieder ein Block, der aus $P \cdot Q$ Blöcken besteht und sich dadurch auszeichnet, dass jeder Prozessor genau einen Block dieses Superblockes speichert. Superblöcke können unvollständig sein, sofern die Größe der Matrix, die Blockgrößen und die Unterteilung der Prozessoren keine multiplikativen Vielfachen sind.

2.1.2 MPI

HPC-Linpack verwendet zur Kommunikation MPI. Die vier wichtigsten Operationen sind:

1. Send: asynchron, aber blockierend bezüglich der verwendeten Puffer, Nachrichten können jedoch zwischengespeichert werden, bis sie empfangen werden. Veranlasst das Senden eine Nachricht vom aktuellen Prozessor an den angegebenen Prozessor.
2. Rec[ei]v[e]: asnychron und blockierend; endet, sobald eine Nachricht vollständig empfangen wurde und wartet ggf. auf das Senden. Empfängt eine Nachricht der Angegebenen Art vom angegebenen oder einem beliebigen Prozessor.
3. B[road]cast: asnychron, blockierend bezüglich der verwendeten Puffer; Sendet an alle Prozessoren der angegebenen Gruppe eine Nachricht.

4. Gather: asnychron, blockierend: Der Aufrufer sammelt von allen angegebenen Prozessoren Daten. Wird z.B. für die Pivotsuche verwendet.

2.1.3 Ergebnisse des Benchmarks

Gemessen wird die Rechenleistung bezüglich der Fließkommaarithmetik. Es werden insgesamt ungefähr $\frac{2}{3}N^3 + 2N^2$ Operationen benötigt. Die Rechenleitung ist dann die Anzahl der Operationen geteilt durch die Zeit für eine Operation. Dies ergibt Fließkommaoperationen pro Sekunde (Flops); das Ergebnis wird meist in GFlops (10^9) angegeben.

2.2 Funktionsweise des parallelen Gaußeliminations-Algorithmus

Der parallele Gaußeliminations-Algorithmus besteht aus zwei Phasen, der Vorwärts- und der anschließenden Rückwärtsubstitution.

1. Phase: Vorwärtsubstitution, n Schritte, $k \in \{1..n\}$

   1. Bestimmung des lokalen Pivot-Elementes: Sei $Co(k)$ die Menge aller Prozessoren, die Teile der Spalte k besitzen. Dann wird das Pivot-Element bezüglich der Spalte k gefunden, indem jeder Prozessor aus $Co(k)$ sein betragsgrößtes Element liefert.

   2. Bestimmung des globalen Pivotelementes: Aus den in Schritt 1. gelieferten Elementen wird das betragsgrößte Element ausgewählt. Der Wert des Pivot-Elementes und der Index der Zeile werden mittels einer Broadcast-Operation an alle Prozessoren gesandt.

   3. Vertauschen der Pivotzeile (im Block): Sei $a^k_{r,k}$ das Pivotelement im $k.$ Durchlauf in Zeile r, die dadurch die Pivotzeile ist. Sei ferner $Ro(r)$ die Menge der Prozessoren, die Teile der Zeile r besitzen. Man beachte, dass k die erste Zeile in der Submatrix des aktuellen Durchlaufes k darstellt. Dann muss eine Tauschoperation vorgenommen werden, sodass die Zeile r und alle ihre Teile mit der Zeile k getauscht werden; die Pivotzeile wird an den Anfang getauscht. Dazu müssen je zwei Prozessoren k_i, r_i aus den Mengen $Ro(k)$ und $Ro(r)$ ihren jeweiligen Teil der Zeile miteinander tauschen. Ggf. sind die Mengen $Ro(k) = Ro(r)$ gleich, gdw. sich beide Zeilen in einem Block befinden. Dann geschieht die Tauschoperation lokal auf jedem Prozessor.

   4. Verteilen der Pivotzeile (im Block): Jeder Prozessor i aus $Ro(k)$ (hier befindet sich nun die Pivotzeile, die aber noch berechnet werden möchte) berechnet die Werte der neuen Pivotzeile, indem er jedes Element mit der Reziproke des Pivotelementes multipliziert. Danach sendet jeder Prozessor i an alle anderen Prozessoren aus der zu i gehörigen Spaltengruppe $Co(i)$ seinen berechneten Teil der Pivotzeile mittels eines Broadcasts.

   5. Berechnung der Eliminationsfaktoren: Jeder Prozessor, der in 4. einen Teil einer Pivotzeile und das Pivotelement $a^(k)_{k,k}$ erhalten hat (die Elemente aus Co(k)), berechnet die Eliminationsfaktoren $l_{i,k}$ für die Elemente i in Spalte k durch Berechnung $l_{i,k} = \frac{a^k_{i,k}}{a^k_{k,k}}$.

   6. Verteilung der Eliminationsfaktoren: Der Eliminationsfaktor der Zeile i wird an jeden anderen Prozessor aus Ro(i) verteilt. Da der Prozessor, der den Eliminationsfaktor berechnet hat, links steht, sendet er ihn quasi an alle rechts von ihm liegenden Prozessoren. Man beachte, dass die Eliminationsfaktoren für einen ganzen Block berechnet werden.

7. Berechnung der Matrizenelemente: Jeder Prozessor aus $Ro(i)$ (s. Schritt 6.) berechnet dann für seine jeweiligen Zeilen i die neuen Zeilen, indem das l_i-fache der Pivotzeile von der i. Zeile subtrahiert wird.

2. Phase: Rückwärtssubstitution, n Schritte, für $k \in \{n..1\}$

   1. Jeder Prozessor aus $Ro(k)$ berechnet die Summe aller Elemente mit Index größer k der Zeile k.
   2. Diese Summen werden in einer globalen Akkumulation summiert und das Ergebnis wird an den Prozessor p gesendet, der $a^n_{k,k}$ besitzt.
   3. p berechnet x_k nach $\frac{1}{a^n_{k,k} \cdot (b^n_k - akku)}$.
   4. p teilt das von ihm berechnete x_k mit den Prozessoren aus $Co(k)$ mittels einer Broadcast-Operation.

2.2.1 Zusätzliche Erklärungen der Funktionen aus HPC-LinPack

- Rekursive Panel-Faktorisierung: In jedem Schritt wird ein Block ausgewählt, der faktorisiert werden soll. Dazu unterteilt LinPack den Block rekursiv in immer kleinere Blöcke, bis das Panel aus höchstens einer vorgegebenen Anzahl von Zeilen besteht. Dann werden die Teilprobleme mit einem Vektor-Matrix-Verfahren gelöst.
- binary-exchange (leave-on-all) reduction: In LinPack geschehen die Pivotsuche, die Zeilenvertauschungen und das Verteilen der Pivotzeile in einem Schritt.
- Rekursive Faktorisierungsstrategie: Es gibt left-looking, right-looking und Crout. Alle faktorisieren die Blöcke auf der Hauptdiagonalen, gehen dabei allerdings unterschiedlich vor. Diese Strategien kommen zweimal vor: In der Matrix-Matrix-Phase, welche die aktuell zu faktorisierende Submatrix unterteilt, und in der Matrix-Vektor-Phase, welche die Teilprobleme berechnet.
  - left-looking: Hierbei wird ein Block faktorisiert, wobei die linken (bereits berechneten) Spalten der unteren Dreiecksmatrix verwendet werden, und dann alle Blöcke in der gleichen Spalte berechnet. Kein Block rechts vom berechneten Block ist bereits berechnet.
  - Crout: Hier sind alle Blöcke bereits berechnet, die sich links, oben und links oben vom zu berechnenden Block befinden. Nach der Berechnung werden alle Blöcke in der gleichen Zeile (rechts) und Spalte (unten) berechnet. Dieses Verfahren eignet sich am besten für Rechner mit hoher Bandbreite und Vektor-Recheneinheiten.
  - right-looking: Der Standard; wie Crout, nur dass nach der Berechnung des Blockes und der Berechnung der Zeile und Spalte auch noch Elemente der Restmatrix transformiert werden. Hier werden meist Matrix-Matrix-Operationen angewendet (bei Blockgröße > 1).
- Strategien bei der Verteilung der Blöcke und Zeilenvertauschungen, angenommen, der Prozess mit der Nummer 0 initiiert die Verteilung; es gibt N Prozessoren:
  - Increasing-ring: Sobald ein Prozess eine Nachricht erhält, sendet er sie an seinen Nachfolger (bestimmt durch seine Nummer).

  - Increasing-ring (modified): Prozess 0 sendet zwei Nachrichten, eine an P1, der diese nicht weitersendet, und eine an P2, der sich wie bei Increasing-ring verhält.
  - Increasing-2-ring: Teilung der Prozessoren nach P0 in zwei gleichgroße Gruppen, die sich intern wie bei Increasing-ring verhalten und deren ersten Prozessor seine Nachricht von P0 erhält.
  - Increasing-2-ring (modified): Hier erhält P1 wieder eine Sonderrolle, sonst wie in Increasing-2-ring.
  - Long (bandwidth reducing): Unterteilung der Nachricht in N Teile, die stückweise verteilt werden und von deren Empfängern mit Nachbarn gegen andere eingetauscht werden.
  - Long (modified): Hier erhält P1 wieder eine Sonderrolle, sonst wie in Long.
- Look-ahead: Ein berechneter Block wird ja verwendet, um die Submatrix zu berechnen. Wenn also der k. Block fertig berechnet wurde, wird er versendet, um die Matrix zu updaten. Danach ist der $k+1$. Block an der Reihe. Look-ahead sorgt dafür, dass genau dieser Block als nächstes fertiggestellt und versandt wird. Bei einem Look-ahead von 1 (manchmal 2) sind die besten Ergebnisse zu erwarten, 0 deaktiviert Look-ahead.
- Update (Berechnung der Submatrix): Da nur der Prozessor, der den Block besitzt, in dem sich die Pivotelemente befinden, die Eliminationsfaktoren berechnen kann, muss dieser Block noch verteilt werden. Dazu gibt es zwei Strategien:
  - Binary-exchange: [...]
  - Long: Analog zu dem Verfahren in „Strategien bei der Verteilung der Blöcke und Zeilenvertauschungen“.
- Gedanken zu möglichen Optimierungen:
  - Gerade bei der Unterteilung der Blöcke muss darauf geachtet werden, dass die entstehenden Konstrukte (Matrizen oder Vektoren) sich mit der Cache-Aritektur der Rechenkerne vertragen. Kleine Blöcke verteilen die Last bestmöglich, vermehren aber die Arbeitsspeicherzugriffe. Matrix-Matrix-Operationen sind auf Superrechnern häufig schneller als Matrix-Vektor-Operationen.
  - Die Strategie Crout, left- & right-looking bestimmen auch die Zugriffsstrategie auf Teile der Matrix. Left nimmt nur große Teile der linken Matrix, Crout alles außer die Submatrizen links oben und rechts unten und right nur rechts unten. Ferner nimmt left keine Blöcke rechts vom zu faktorisierenden Block; right und Crout schon.

2.3 Die Parameter-Datei

Die Eingaben der Parameter erfolgen in der Datei „HPL.dat“. Diese Datei dient als Eingabedatei für einen LinPack-Lauf.

- Zeile 1 und 2 werden ignoriert und enthalten meist Text.
- Zeile 3 gibt die optionale Datei der Ausgabeumleitung an.
- Zeile 4 gibt die Ausgabekanäle an.

Es folgen die wichtigen Parameter.

- Zeile 5 gibt die Anzahl der Durchläufe mit verschiedenen Problemgrößen an, deren genaue Größe in Zeile 6 spezifiziert wird.
- Zeile 6 gibt genau die n Problemgrößen an, die in Zeile 5 definiert wurden. Für jeden dieser n Werte wird ein LinPack-Durchlauf mit eben dieser Problemgröße gestartet.
- Zeile 7 gibt die Anzahl der zu benutzenden Blockgrößen an, die in Zeile 8 spezifiziert werden.
- Zeile 8 gibt die n Blockgrößen an, die in Zeile 7 angegeben wurden. Für jede dieser Blockgrößen wird ein LinPack-Durchlauf gestartet.
- Zeile 9 gibt den Abbildungsmodus von MPI-Prozessen auf die Rechenknoten an. Empfohlen wird row-major, d.h. Zeilen in der Matrix bilden im Hauptspeicher zusammenhängende Einheiten, wohingegen Einträge in einer Reihe über den Speicher verteilt sind.
- Zeile 10 gibt die Anzahl n der Abmessungen (P * Q) des Rechenknotengitters an, wobei die konkreten Abmessungen in Zeile 11 spezifiziert werden.
- Zeile 11 gibt die Größe P der in Zeile 10 angegebenen Abmessungen in einem Gitter an, Zeile 12 die Größe Q. P*Q ergibt die gesamte Anzahl an verwendeten Rechenknoten.
- Zeile 13 gibt den Grenzwert des Residuums an, d.h. die maximal noch gültige Abweichung des tatsächlichen Ergebnisses. Dieser Wert erklärt gegebenenfalls Durchläufe als fehlerhaft und hat nichts mit der Laufzeit zu tun. Ein negativer Wert überspringt alle Prüfungen. LinPack prüft 3 Prädikate (eps bezeichnet die kleinstmögliche Fließkommagenauigkeit der Maschine, ||A|| bezeichnet die Matrixnorm, d.h. das Maximum aller Summen über die Beträge der Elemente einer Spalte.):
  1. $\frac{||Ax-b||_{oo}}{eps \cdot ||A||_1 \cdot N}$
  2. $\frac{||Ax-b||_{oo}}{eps \cdot ||A||_1 \cdot ||x||_1}$
  3. $\frac{||Ax-b||_{oo}}{eps \cdot ||A||_{oo} \cdot ||x||_{oo}}$

Es folgen spezielle algorithmische Einstellungen:

- Zeile 14 gibt die Anzahl der (ggf. rekursiven) Panel-Faktorisierungs-Strategien für die Unterteilung in Submatrizen an, die in Zeile 15 spezifiziert werden.
- Zeile 15 gibt die Art der Panel-Faktorisierungen an. Dabei ist 0 links-, 2 rechts-schauende Faktorisierung und 1 Crout-Faktorisierung.
- Zeile 16 gibt die maximale rekursive Tiefe der Faktorisierungen an, bis die in Zeile 17 angegebene Höchstanzahl der Spalten im aktuellen Block erreicht ist.
- Zeile 17 gibt die maximale ("less than or equal to NBMIN") Spaltenanzahl im aktuellen Panel an, bei der der Unterteilungsvorgang abgebrochen werden soll.
- Zeile 18 gibt die Anzahl der Teile an, die in jedem Unterteilungsschritt entstehen.
- Zeile 19 gibt an, wie viele Teile jeweils in einem Unterteilungsschritt entstehen. Für normale Rekursion mit Matrix-Vektor-Operationen ist z.B. Zeile 19 (NDIV) = 2, Zeile 17 (NBMIN) = 1.

- Zeile 20 gibt die Anzahl der Faktorisierungs-Strategien der entstehenden Teilprobleme an, die in Zeile 21 spezifiziert werden.
- Zeile 21 gibt die Art der Faktorisierungen für die Teilprobleme an. Dabei ist 0 links-, 2 rechts-schauende Faktorisierung und 1 Crout-Faktorisierung.
- Zeile 22 gibt die Anzahl der Broadcast-Strategien an, die in Zeile 23 spezifiziert werden.
- Zeile 23 gibt die Art der Broadcast-Strategien an. Dabei ist 0 = Increasing-ring, 1 = Increasing-ring (modified), 2 = Increasing-2-ring, 3 = Increasing-2-ring (modified), 4 = Long, 5 = Long (modified).
- Zeile 24 gibt die Anzahl der Look-ahead-Tiefen an, die in Zeile 25 spezifiziert werden.
- Zeile 25 gibt die Look-ahead-Tiefen an. 0 deaktiviert Look-ahead, gute Werte sind 1 oder 2.
- Zeile 26 gibt die Art der Verteilung von Blöcken zur Berechnung der Submatrix an. 0 ist binary-exchange, 1 ist Long, 2 eine Mischung aus 0 und 1.
- Zeile 27 gibt die Anzahl der Spalten an, die erreicht werden müssen. Wird in HPC-LinPack nicht verwendet!
- Zeile 28 gibt an, ob die Spalten in Böcken der oberen Dreiecksmatrix transponiert gespeichert werden soll (0) oder nicht (1).
- Zeile 29 gibt an, ob die Zeilen in Böcken der unteren Dreiecksmatrix transponiert gespeichert werden soll (0) oder nicht (1).
- Zeile 30 gibt an, ob Equilibration für das Long-Verfahren aus Zeile 26 benutzt werden soll (1) oder nicht (0).
- Zeile 31 gibt die Speicheranordnung in Bytes für Double an (Normal: 8).

3 Ergebnisse

Die folgenden Parameterwerte sind Empfehlungen, mit denen die besten Ergebnisse bezüglich der Leistung erzielt wurden (außer PFACT und RFACT, auf die die Wahl zugunsten der kleinsten Schwankungen fiel). Sie sind gleichzeitig die Standardparameter, die bei den Leistungs- und Energiemessungen verwendet wurden. Diejenigen Parameter, die bei Messungen variiert wurden, sind jeweils extra aufgeführt.

- N = 20000
- NB = 16
- PMAP = Row-major process mapping
- P = 4
- Q = 6
- PFACT = Crout
- NBMIN = 4
- NDIV = 2

- RFACT = Crout
- BCAST = 1ringM
- DEPTH = 0
- SWAP = Mix (threshold = 64)
- L1 = transposed form
- U = transposed form
- EQUIL = yes
- ALIGN = 8 double precision words

Jeder Testlauf wurde fünfmal ausgeführt und das arithmetische Mittel über die Ergebniswerte gebildet. Ferner sind folgende Aspekte zu beachten:

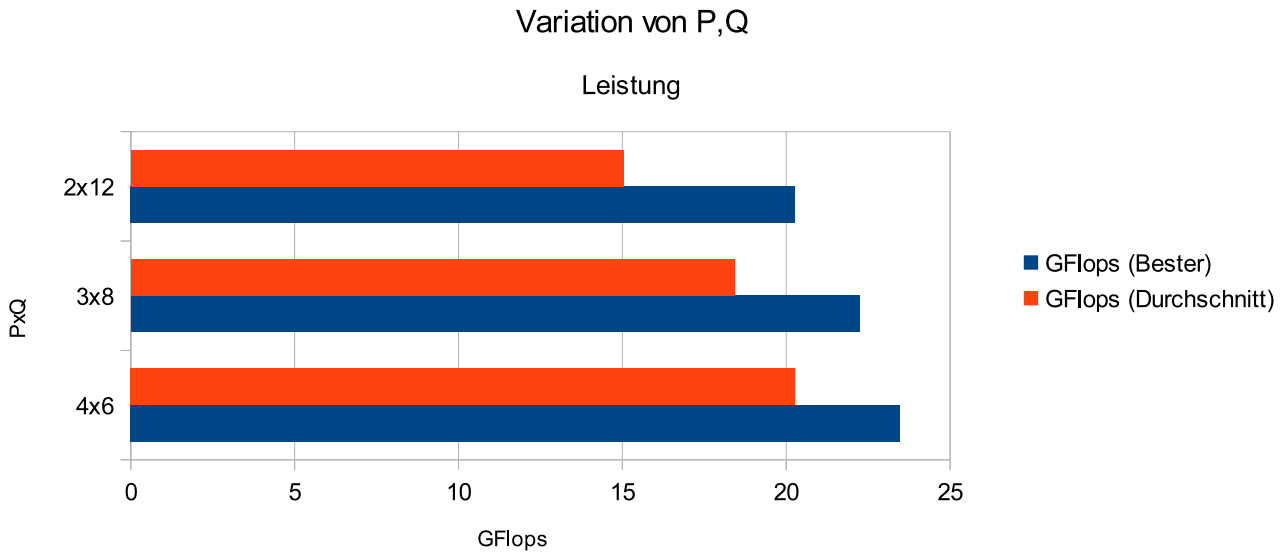
- Sofern nicht anders angegeben, wurden die Messungen auf einem Cluster der Größe 1 ausgeführt.
- Ein Cluster besitzt 2 Rechenknoten mit je 12 Rechenkernen, d.h. es gilt für alle Kombinationen von P,Q: $P \cdot Q = 24$.
- Die Größe der Matrix im Speicher ist $20000^2 \cdot 8$ Byte $= 3200000000$ Byte $\approx 2,98$ GB.
- Bei Messungen auf einem Cluster der Größe 5 gibt es $5 \cdot 24 = 120$ Rechenknoten.
- Bei Cluster der Größe 5 sind standardmäßig P = 12, Q = 10 und NB = 42.
- Alle Knoten sind gleich schnell.

3.1 Leistungsmessungen

Diese Messungen wurden angefertigt, um die Parameter zu finden, die zur größten Rechenleistung führen. Der Energieverbrauch wurde hier außer Acht gelassen; er folgt im nächsten Teil. Ferner wurde versucht, eine Einschätzung der Wirkung auf die Rechenleistung abzugeben.

3.1.1 Variation von P und Q

Hier wurden die Parameter P und Q verändert und für jede gültige Kombination von P und Q und $NB \in \{8, 16, 32\}$ die Leistung gemessen. Schließlich wurden gemittelter und bester Wert verwendet.

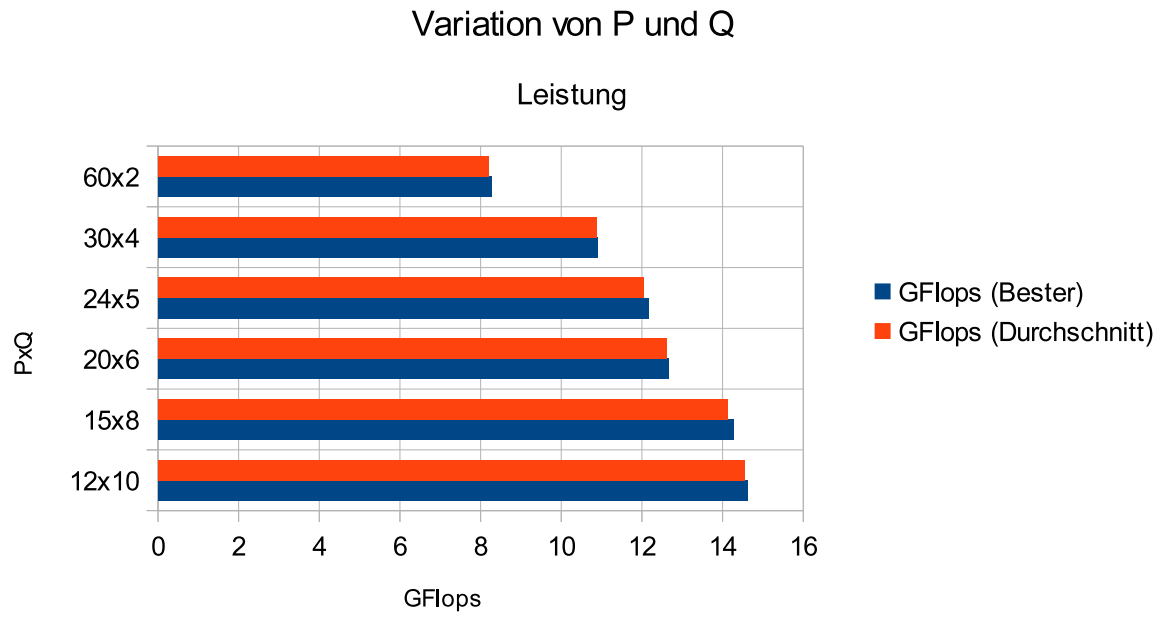


Beobachtung: Die besten Ergebnisse entstanden, wenn P und Q ein möglichst quadratisches Rechteck aufspannen.

Einschätzung: Die Auswirkungen von nicht optimalen Werten für P und Q sind gravierend. Sie senken die Durchschnittsleitung um 30%.

3.1.2 Variation von P und Q auf Cluster der Größe 5

Hier wurde wie oben vorgegangen, nur dass ein Cluster der Größe 5 verwendet wurde.

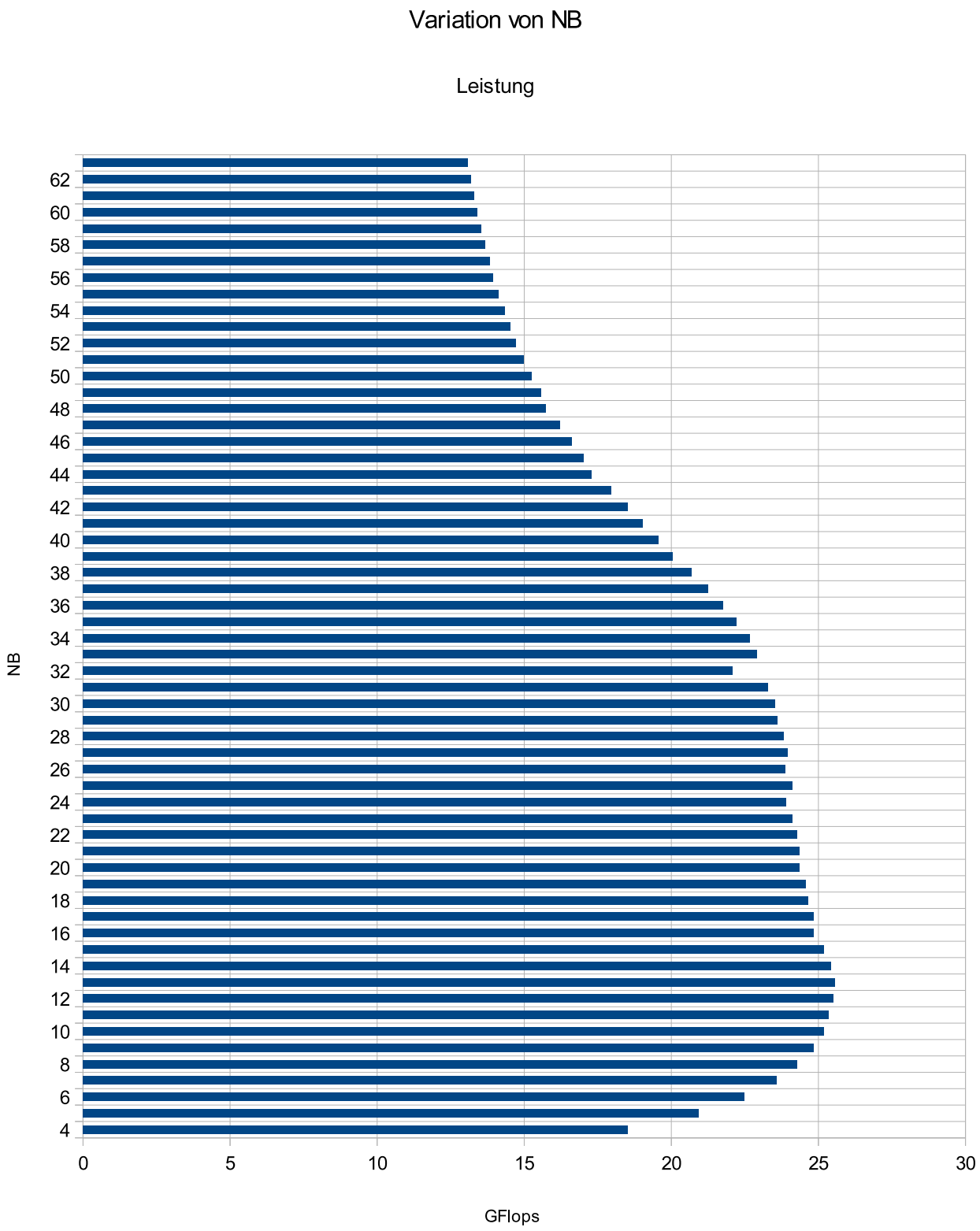


Beobachtung: Die besten Ergebnisse entstanden wieder, wenn P und Q ein möglichst quadratisches Rechteck aufspannen.

Einschätzung: Die Auswirkungen von nicht optimalen Werten für P und Q sind noch gravierender als bei einem Cluster. Sie senken die Durchschnittsleitung um über 40%.

3.1.3 Variation von NB

Hier wurde für $NB \in \{4, .., 32\}$ die Leistung gemessen, allerdings ohne Wiederholung der Messungen, weshalb die Ergebnisse Rauschen enthalten können.



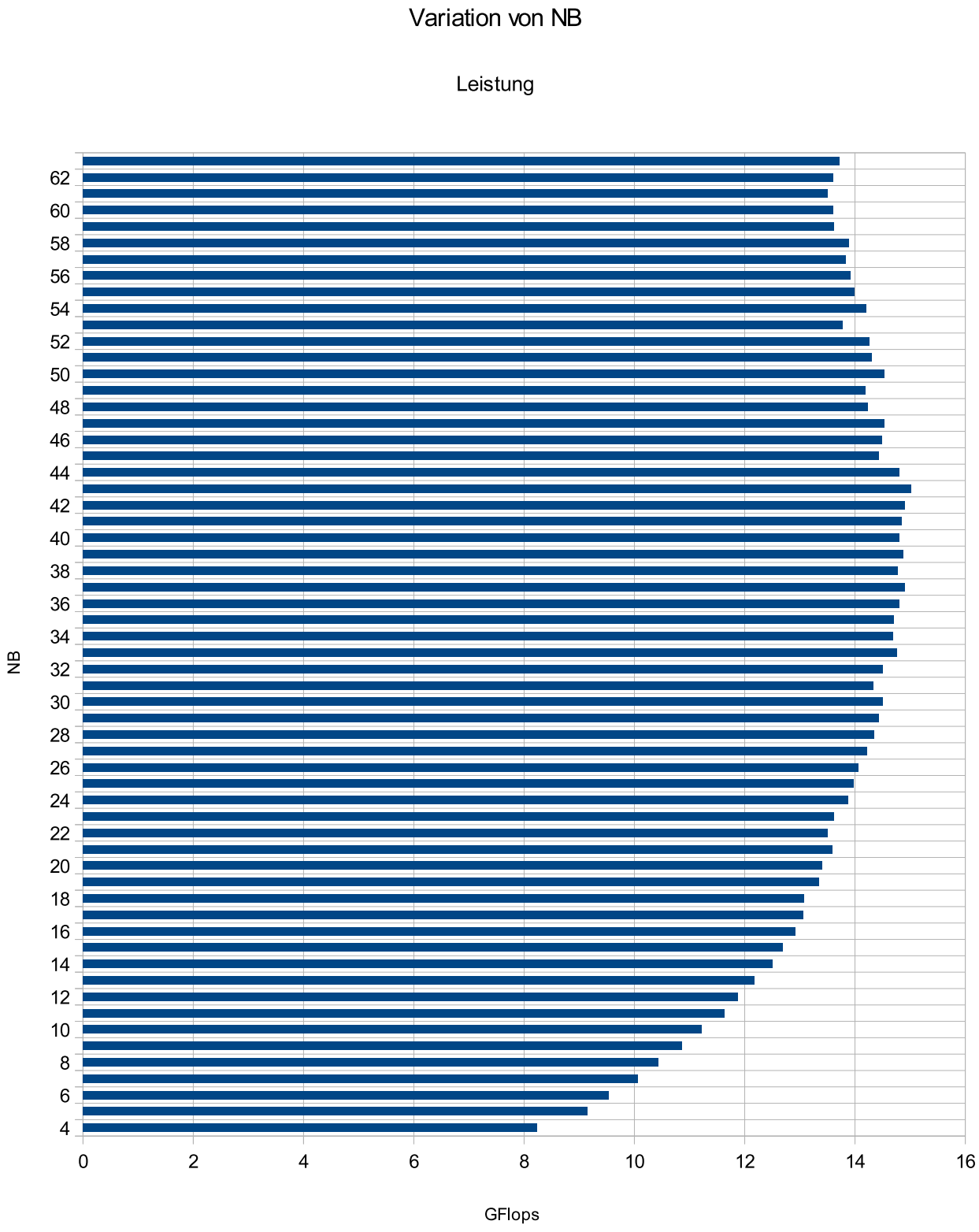
Beobachtung: Die besten Ergebnisse liegen bei NB $= 13$, allerdings weisen nahe Werte eine

ähnlich gute Leistung auf.

Einschätzung: Die Auswirkungen von nicht optimalen Werten für NB sind geringer als bei der Variation von P und Q, aber noch spürbar. Sie senken die Durchschnittsleitung um über 10% für leicht abweichende, aber um bis zu 20% für schlechte, zu kleine Wahl von NB. Für zu großes NB ist der Leistungsverlust sogar noch größer, hier liegt der Verlust bei 50%.

3.1.4 Variation von NB auf Cluster der Größe 5

Hier wurde wie oben vorgegangen, nur dass ein Cluster der Größe 5 verwendet wurde, aber ebenfalls ohne Wiederholung der Messungen.

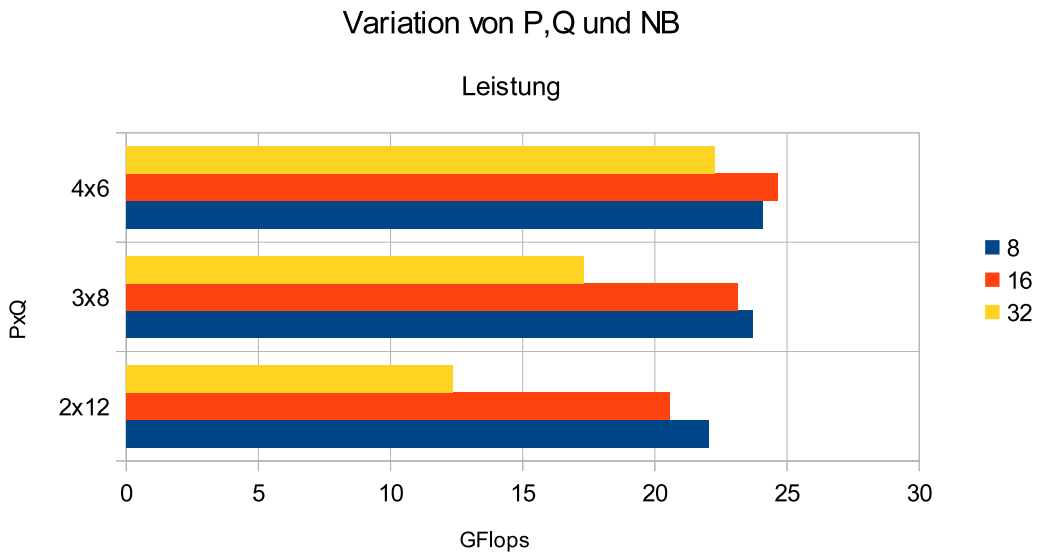


Beobachtung: Auf einem Cluster der Größe 5 ist die Leistungswerte schlechter als auf einem Cluster der Größe 1, der Energieverbrauch für jeden einzelnen Teil liegt aber bei dem einer Messung mit einem Cluster der Größe 1. Damit ist der Gesamtverbrauch fünfmal höher. Die beste Blockgröße ist 42, gerade Werte sind leicht besser als ungerade.

Einschätzung: Die Auswirkungen von nicht optimalen Werten für NB sind analog zu denen bei einem Cluster der Größe 1.

3.1.5 Variation von NB und P,Q

Hier wurden neben den Parametern P und Q auch den Parameter NB verändert. Dann wurde für jede gültige Kombination von P und Q und $NB \in \{8, 16, 32\}$ die Leistung gemessen und der gemittelte Wert verwendet.

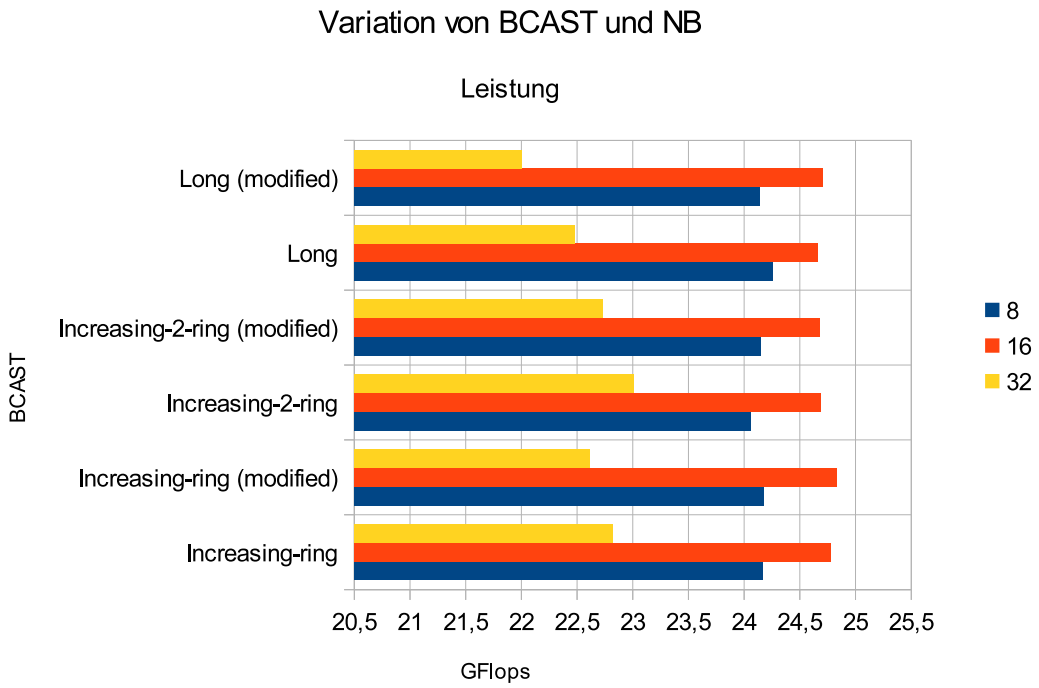


Beobachtung: Die besten Ergebnisse entstehen, indem P und Q möglichst nahe am quadratischen Ideal liegen und NB klein ist. Hier ist NB $=$ 16 ideal. Bei kleineren Werten ist die Leistung ein wenig, bei größeren wesentlich schlechter. Ferner fällt auf, dass NB $=$ 8 optimal ist, sobald P und Q sich vom quadratischen Ideal entfernen.

Einschätzung: Die Auswirkungen sind in erster Linie abhängig von der Wahl von P und Q. Bei guter Wahl verschlechtert die suboptimale Wahl von NB das Ergebnis um 5%, bei schlechter Wahl von P und Q hingegen um über 40%.

3.1.6 Variation von BCAST und NB

Hier wurden neben dem Parameter NB auch BCAST verändert. Dann wurde für $NB \in \{8, 16, 32\}$ und BCAST $\in \{0, 1, 2, 3, 4, 5\}$ die Leistung gemessen und der gemittelte Wert verwendet.

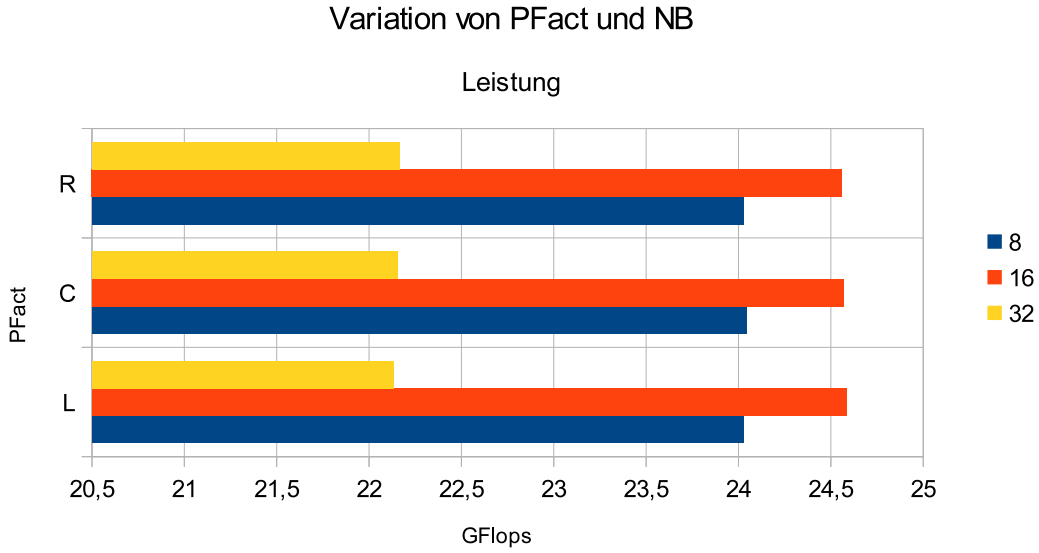


Beobachtung: Die besten Ergebnisse entstehen unabhängig von der verwendeten Broadcast-Strategie mit einer Blockgröße von 16. Wieder sind zu kleine Blockgrößen besser als zu große, ideal ist wieder NB = 16.

Einschätzung: Die Auswirkungen richten sich in erster Linie nach der Wahl von P und Q, die Broadcast-Strategie ist auf einem Cluster der Größe 1 nur für schlechte Wahl von P und Q sichtbar, dann liegt sie bei 5%.

3.1.7 Variation von PFact und NB

Hier wurden die Parameter PFact (äußere Faktorisierungsstrategie) und NB verändert. Dabei bedeutet L: left-looking , R: right-looking und C: Crout.

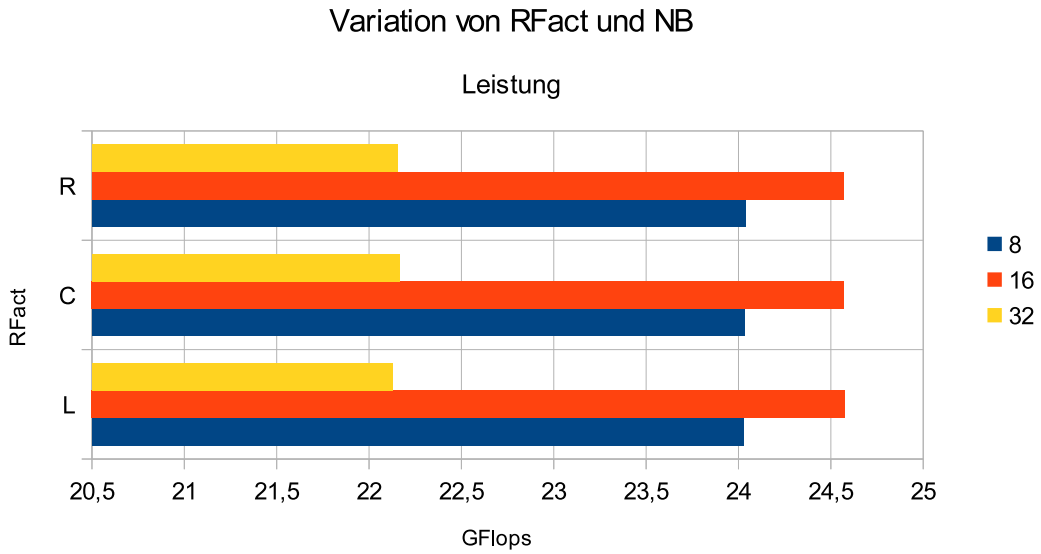


Beobachtung: Die besten Ergebnisse entstehen unabhängig von der äußere Faktorisierungsstrategie mit einer Blockgröße von 16. Wieder sind zu kleine Blockgrößen besser als zu große, ideal ist wieder NB = 16.

Einschätzung: Die Auswirkungen sind analog zu denen der Wahl der Blockgröße.

3.1.8 Variation von RFact und NB

Hier wurden die Parameter RFact (innere Faktorisierungsstrategie) und NB verändert. Dabei bedeutet wieder L: left-looking , R: right-looking und C: Crout.

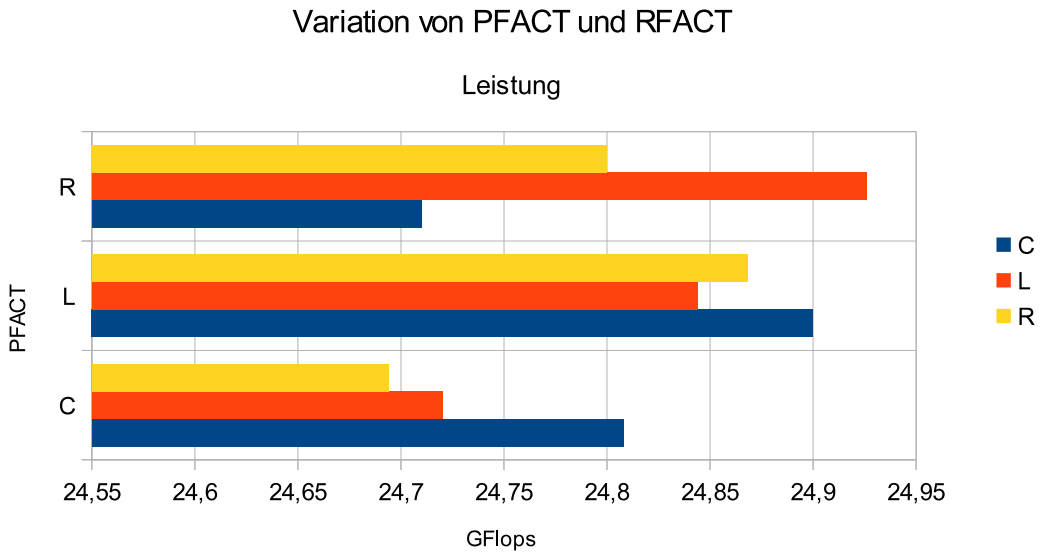


Beobachtung: Die besten Ergebnisse entstehen wie bei der Messung zu PFACT unabhängig von der inneren Faktorisierungsstrategie mit einer Blockgröße von 16. Ebenfalls sind zu kleine Blockgrößen besser als zu große, ideal ist wieder NB = 16.

Einschätzung: Die Auswirkungen sind analog zu denen der Wahl der Blockgröße.

3.1.9 Variation von PFact und RFact

Hier wurden die Parameter PFact (äußere Faktorisierungsstrategie) und RFact (innere Faktorisierungsstrategie) verändert.

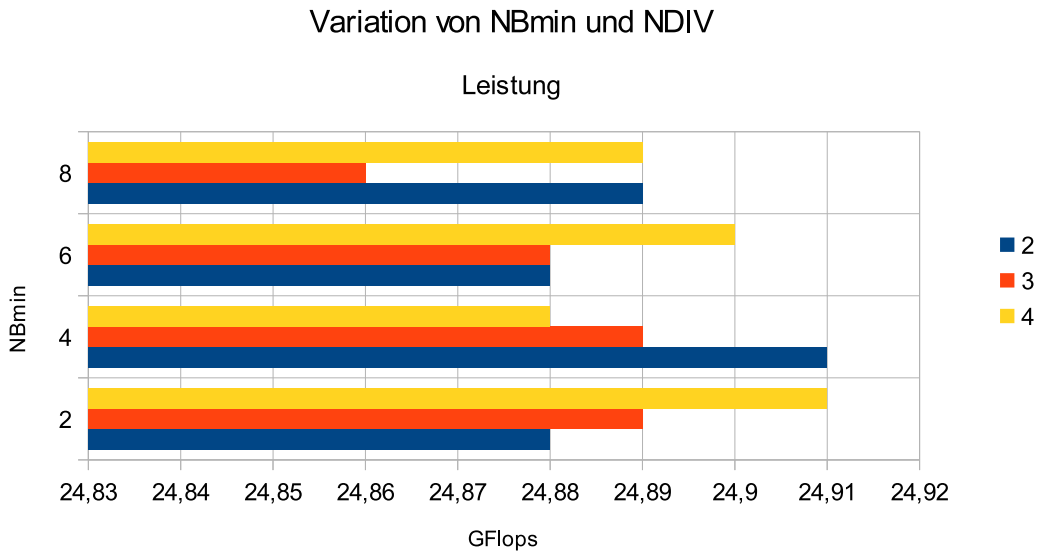


Beobachtung: Die besten Ergebnisse entstehen mit einer Kombination von PFact = L, RFact ≠ L oder mit PFact = RFact = C.

Einschätzung: Die Leistungseinbußen einer schlechten Wahl von PFACT und RFACT liegen unter 1%.

3.1.10 Variation von NBMIN und NDIV

Hier wurden die Parameter NBMIN und NDIV verändert und für NBMIN $\in \{2, 4, 6, 8\}$ und NDIV $\in \{2, 3, 4\}$ die Leistung gemessen und der gemittelte Wert verwendet.



Beobachtung: Die besten Ergebnisse entstehen mit (NBMIN, NDIV) $= \{2, 4\}$.

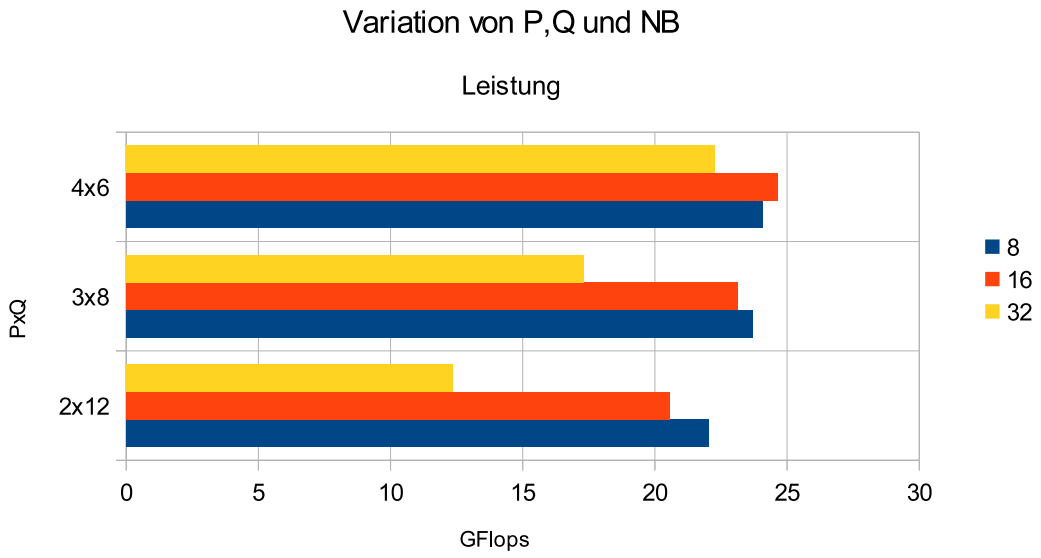
Einschätzung: Die Leistungseinbußen einer schlechten Wahl von NBMIN und NDIV liegen weit unter 1%.

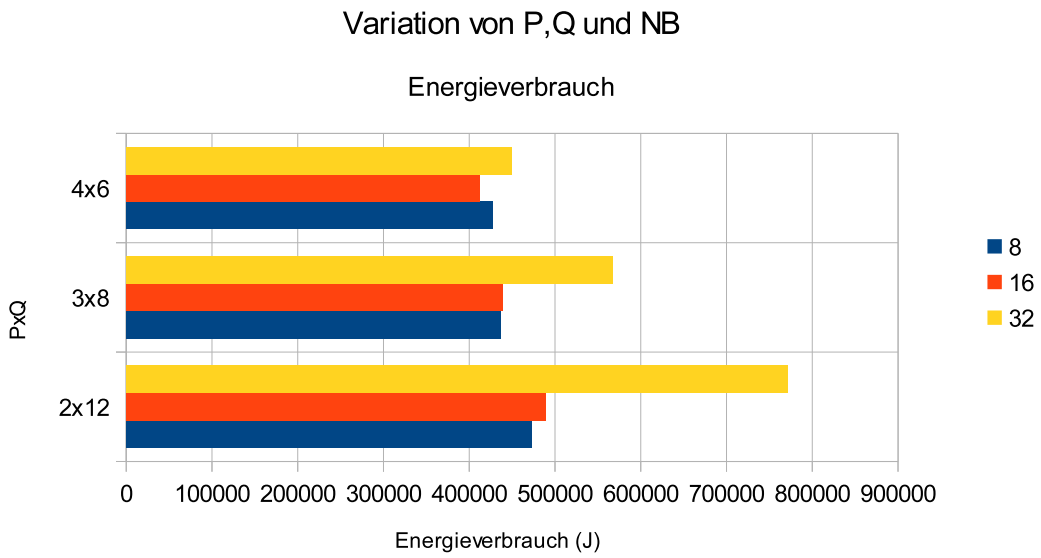
3.2 Energiemessungen

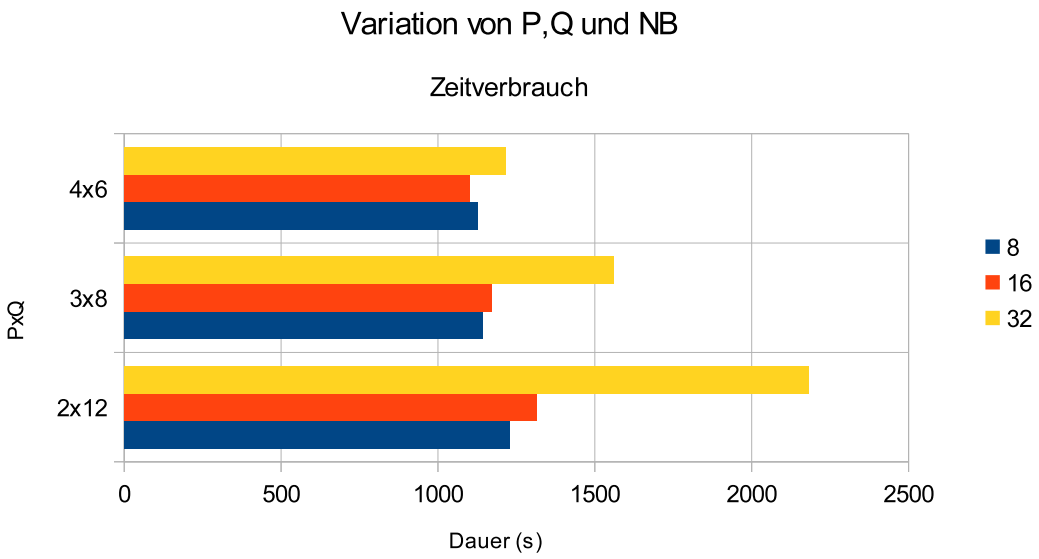
Für die Energiemessungen gelten die gleichen Standard-Parameter wie für die Leistungsmessungen. Für jede Messung ist die Rechenleistung (zum Vergleich), die verbrauchte Energie, die Dauer, die durchschnittliche Leistung und schließlich die Energieeffizienz (in $\frac{MFlops}{J}$) aufgeführt.

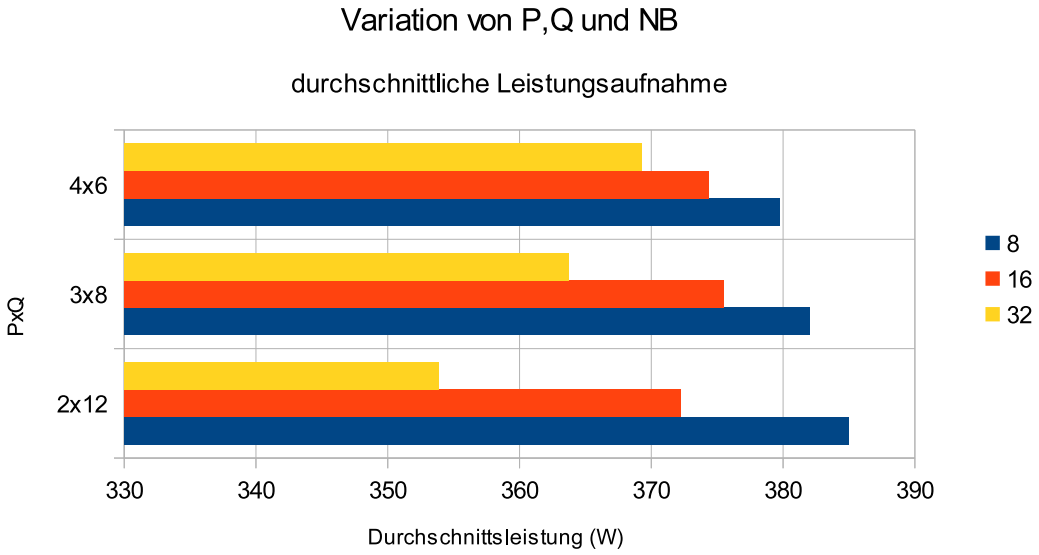
3.2.1 Variation von NB, P und Q

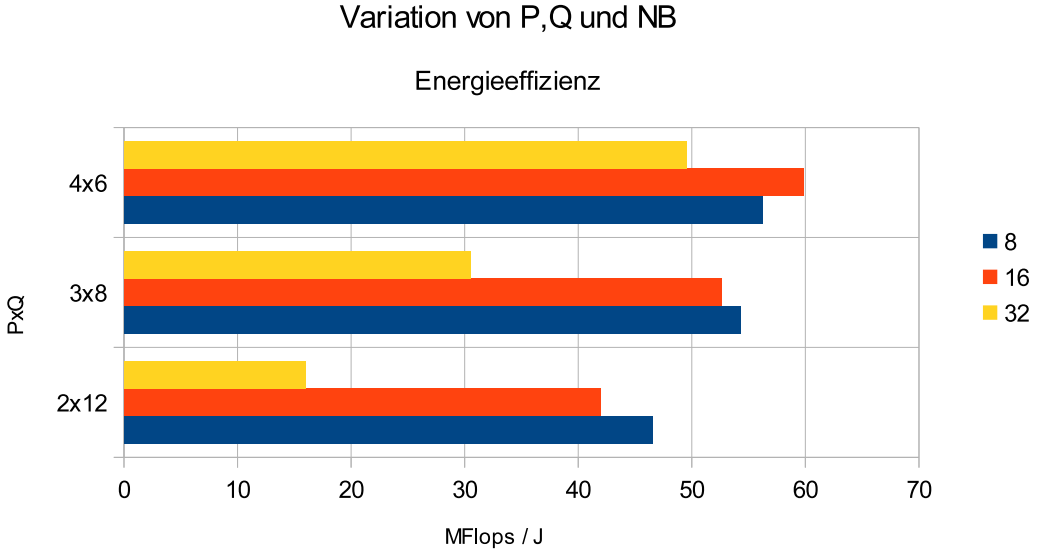
Hier wurden die Parameter P und Q sowie NB verändert und Messungen für jede gültige Kombination von P und Q und NB $\in \{8, 16, 32\}$ durchgeführt.






Variation von P,Q und NB
Zeitverbrauch
4x6
3x8
2x12
PxQ
0
500
1000
1500
2000
2500
Dauer (s)
8
16
32


Variation von P,Q und NB
durchschnittliche Leistungsaufnahme
4x6
3x8
2x12
PxQ
330
340
350
360
370
380
390
Durchschnittsleistung (W)
8
16
32

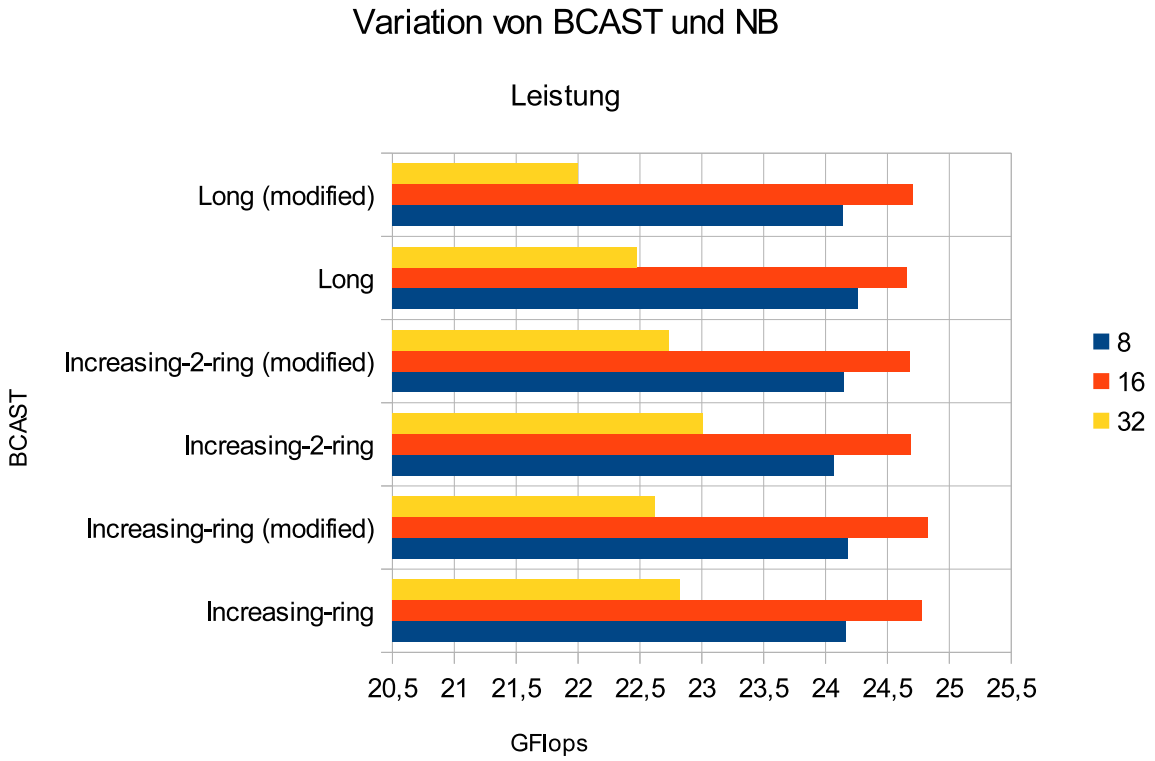

Variation von P,Q und NB
Energieeffizienz
4x6
3x8
2x12
PxQ
0
10
20
30
40
50
60
70
MFlops / J
8
16
32

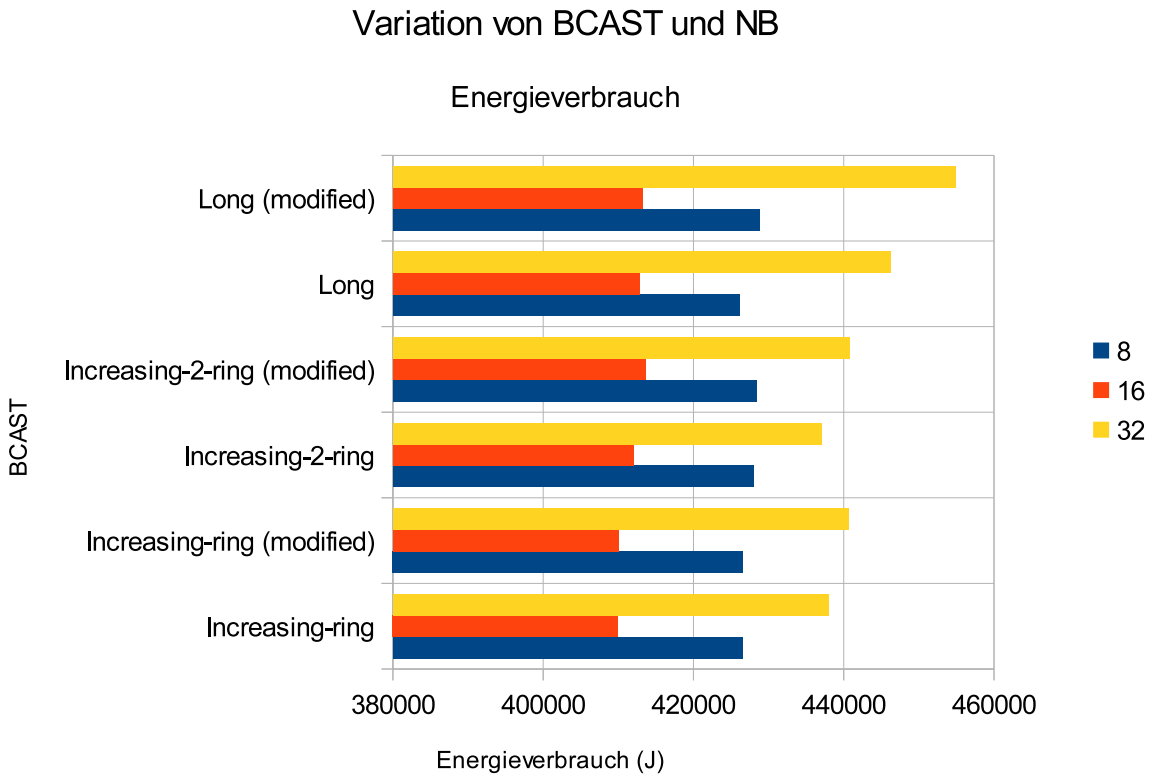
Beobachtung: Der Energieverbrauch steigt um 20% bei sehr schlechter Wahl von P und Q, während die Leistung um 15% sinkt. Bei zusätzlich schlechter Wahl von NB liegen die Leistungseinbußen bei 40%, bei guter bei 5%. Bei zu großen Werten für NB sinkt die durchschnittliche Leistungsaufnahme, bei zu kleinen steigt sie.

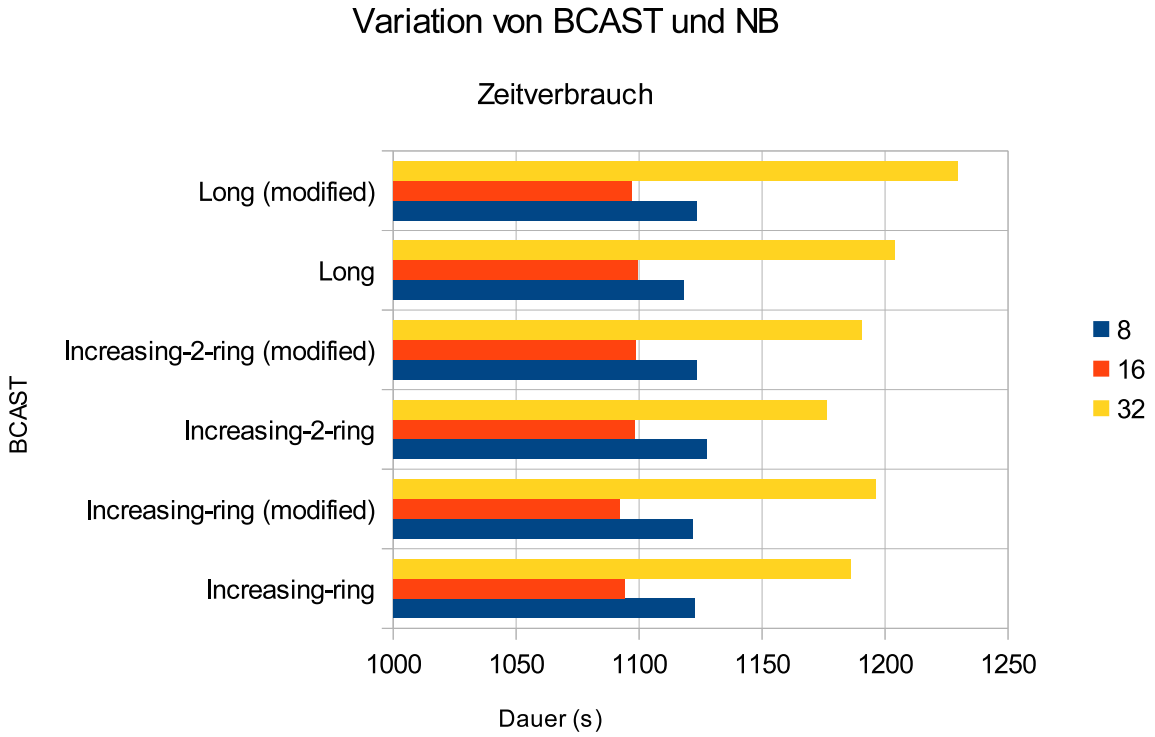
Einschätzung: Je weiter die Parameter P und Q vom quadratischen Ideal entfernt liegen, desto stärker fällt die Wirkung der Wahl von NB aus. Schlechte Wahl von P und Q erhöhen die verbrauchte Energie und die benötigte Zeit bei zu großem NB um 40%. Bei zu kleinen Werten für NB liegen die Einbußen der Rechenleitung und des Energieverbrauches bei 5%.

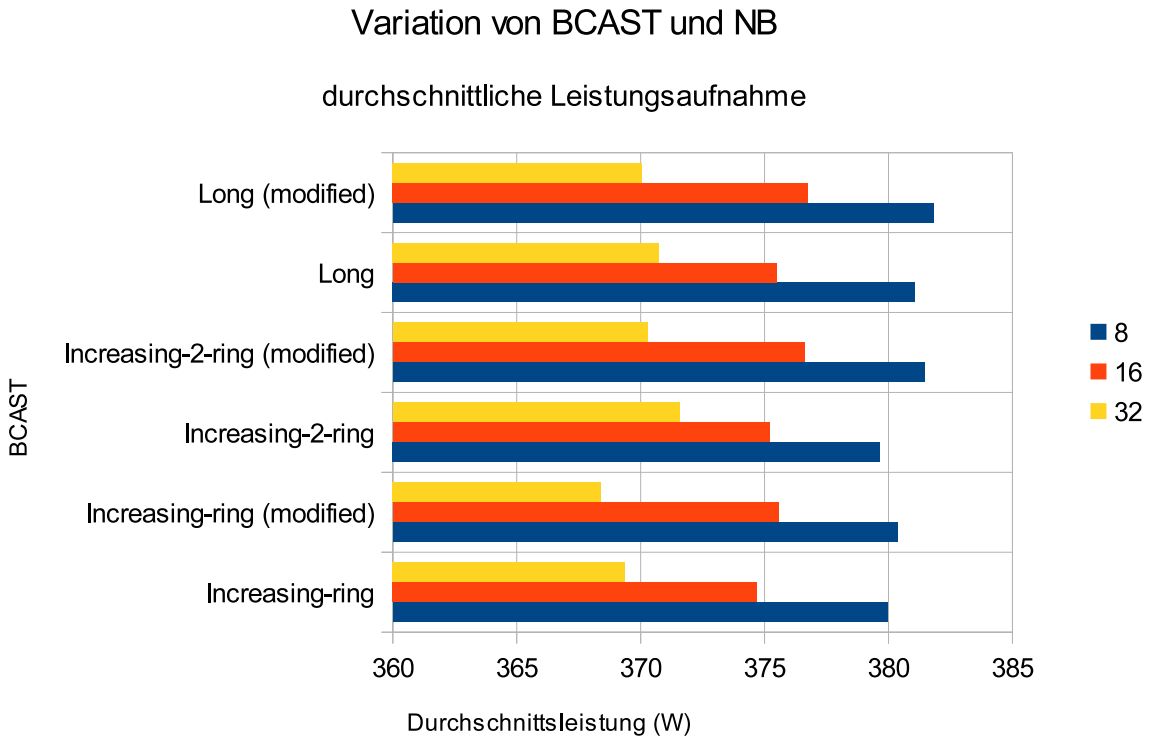
3.2.2 Variation von NB und BCAST

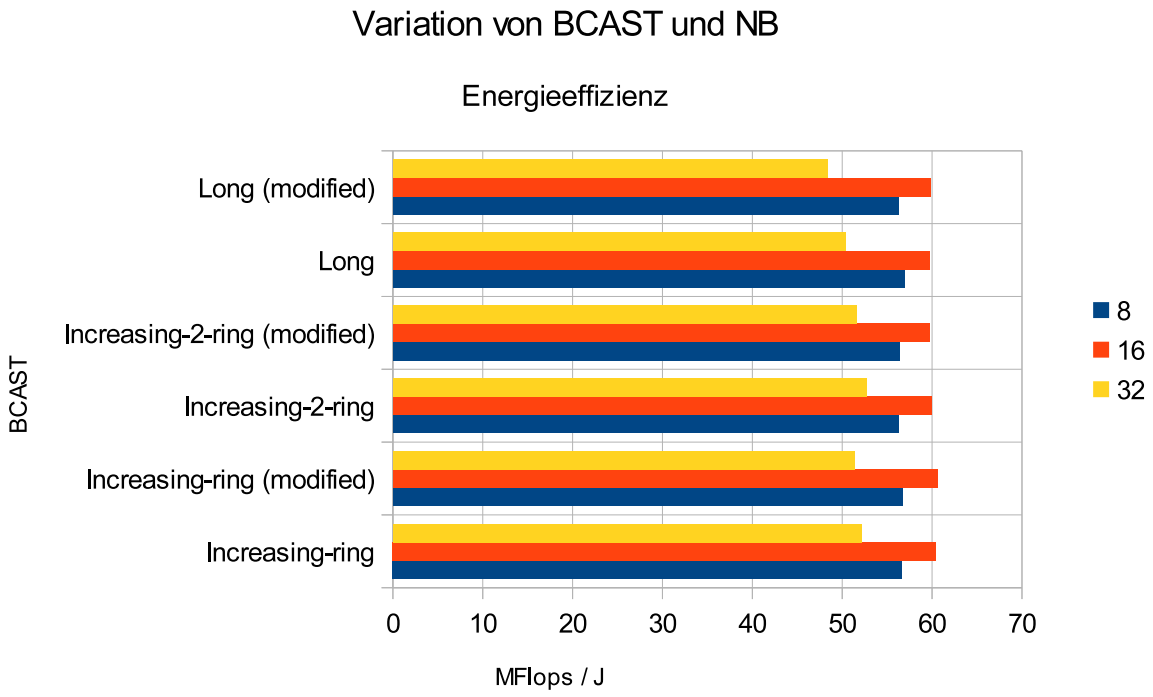
Hier wurden die Parameter NB sowie BCAST verändert und Messungen für NB $\in \{8, 16, 32\}$ und BCAST $= \{$Increasing-ring, Increasing-ring (modified), Increasing-2-ring, Increasing-2-ring (modified), Long, Long (modified)$\}$ durchgeführt.




Variation von BCAST und NB
Energieverbrauch
Long (modified)
Long
Increasing-2-ring (modified)
Increasing-2-ring
Increasing-ring (modified)
Increasing-ring
BCAST
380000
400000
420000
440000
460000
Energieverbrauch (J)
8
16
32


Variation von BCAST und NB
Zeitverbrauch
Long (modified)
Long
Increasing-2-ring (modified)
Increasing-2-ring
Increasing-ring (modified)
Increasing-ring
BCAST
1000
1050
1100
1150
1200
1250
Dauer (s)
8
16
32





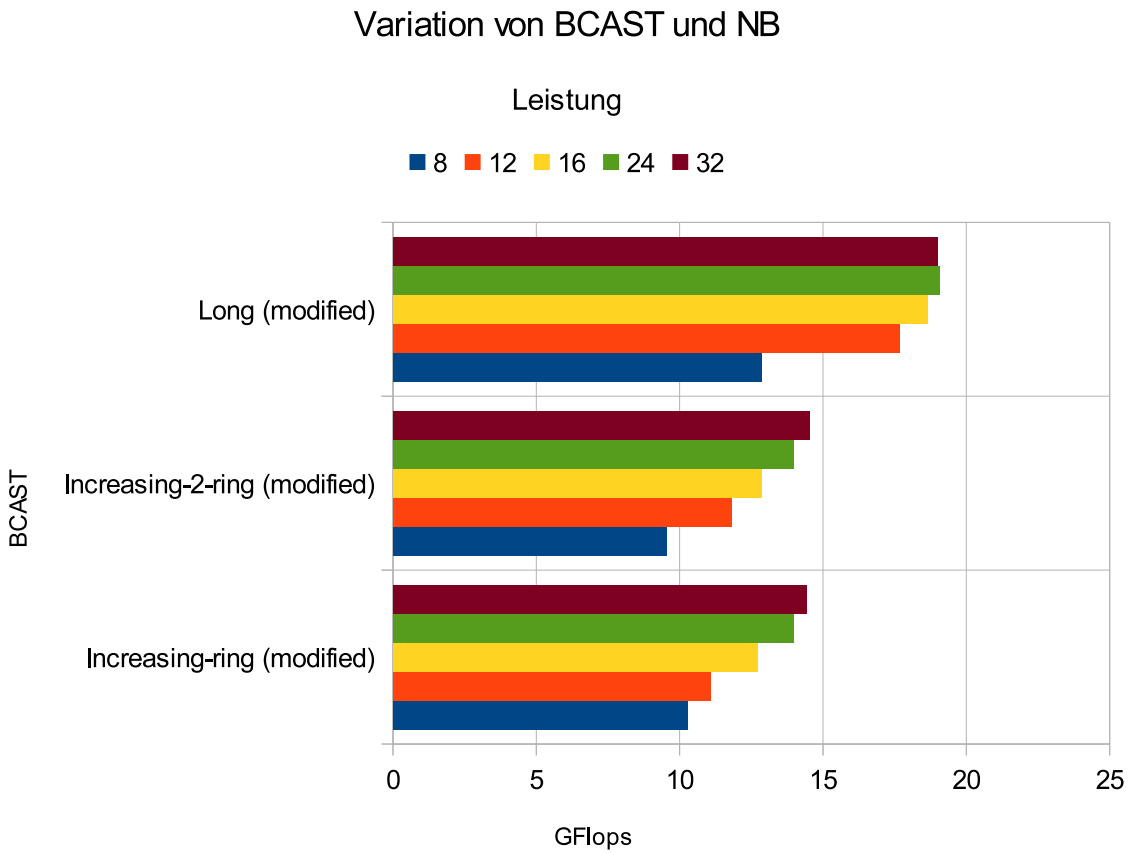
Beobachtung: Aus einem Cluster der Größe 1 spielt die Broadcast-Strategie für optimale Blockgröße (NB = 16) eine untergeordnete Rolle. Hier ist „inreasing-ring (modified)“ am besten. Bei schlechter Wahl von NB verschärfen sich die Unterschiede: „inreasing-2-ring“ ist am besten, „long“ hat den höchsten Energiebedarf, aber schlechtere Leistung.

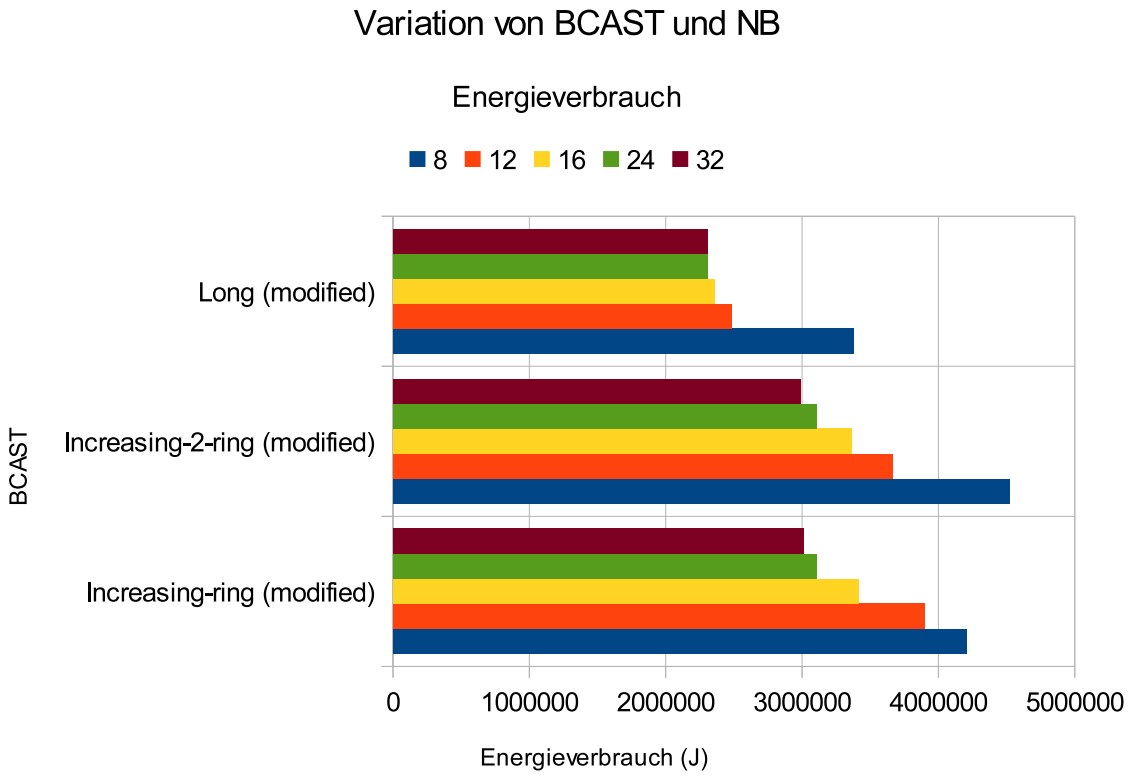
Einschätzung: Wieder führen zu kleine Werte für NB zu besserer Rechenleitung und weniger verbrauchter Energie als zu große Werte. Der Energieverbrauch steigt bei schlechter Wahl von NB fast unabhängig um 2%. Damit ist die Leistung stärker betroffen als der Energiever-

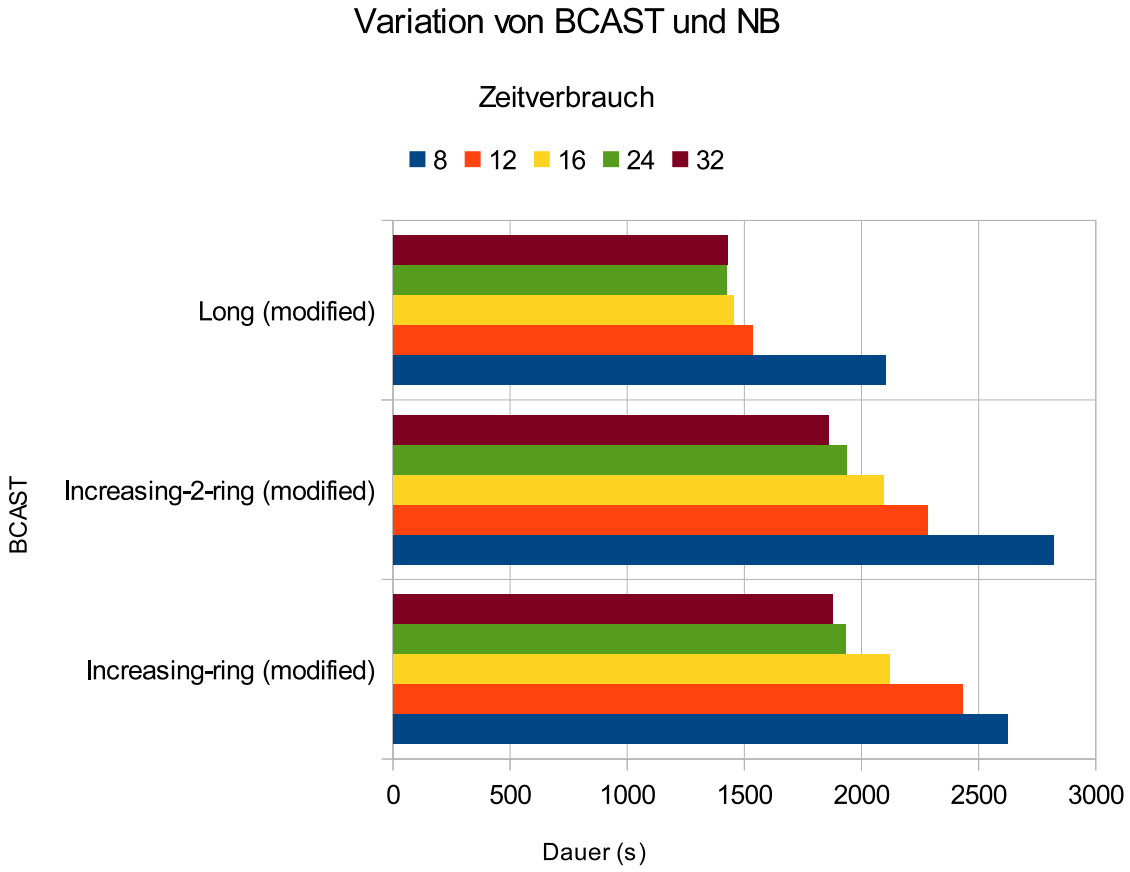
brauch.

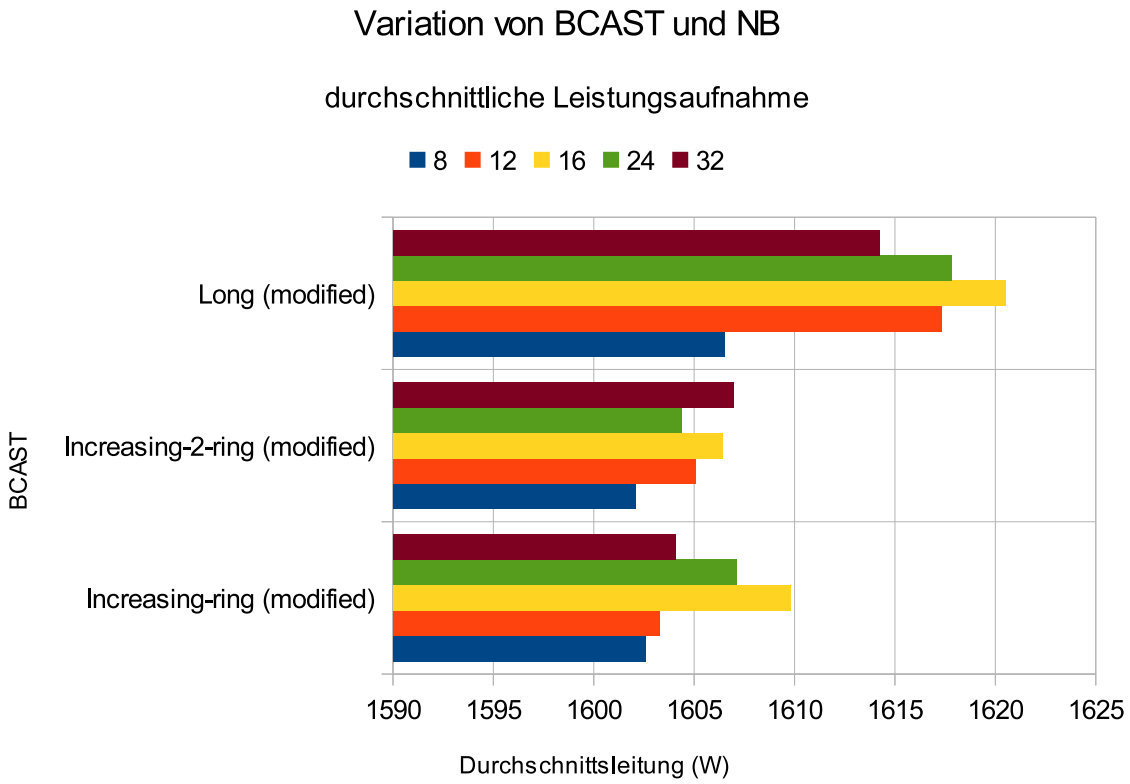
3.2.3 Variation von NB und BCAST auf Cluster der Größe 5

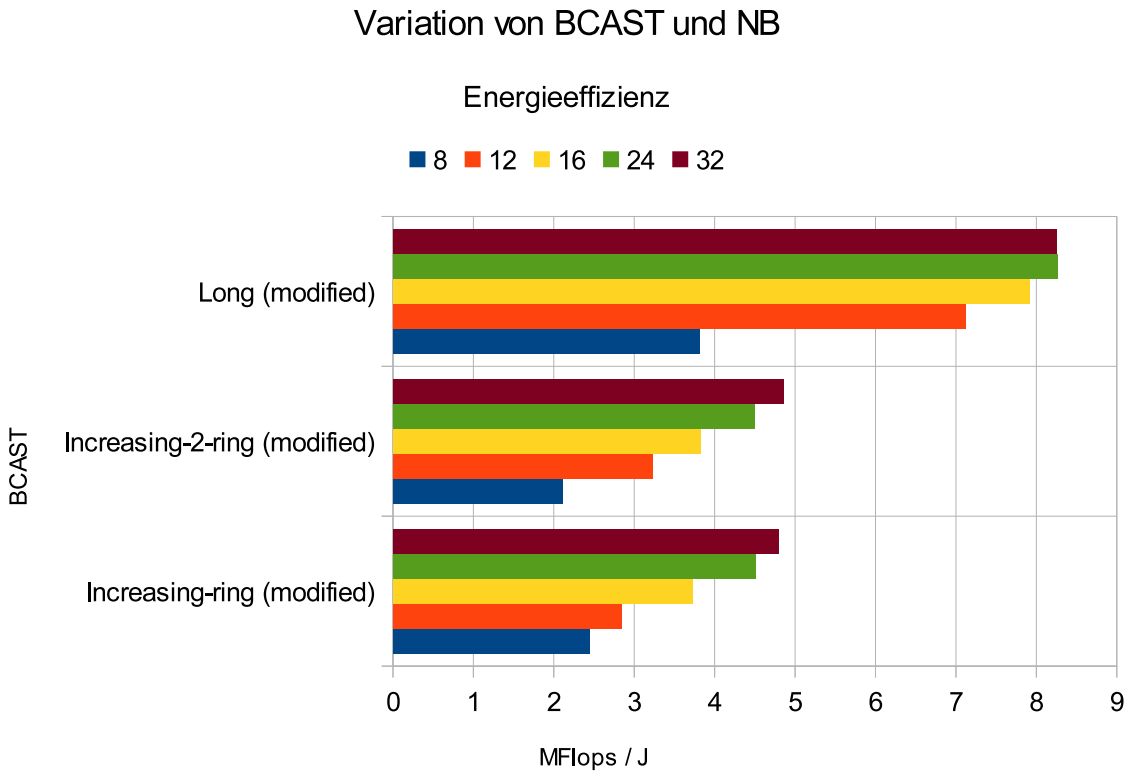
Hier wurde ein Cluster der Größe 5 verwendet sowie NB $\in \{8, 12, 16, 24, 32\}$ und BCAST $=$ {Increasing-ring (modified), Increasing-2-ring (modified), Long (modified)} gewählt. Ferner wird als Zeitverbrauch der Durchschnitt, für Energieverbrauch und Leistungsaufnahme die Summe aller Rechenknoten angenommen.




Variation von BCAST und NB
Energieverbrauch
8
12
16
24
32
BCAST
Long (modified)
Increasing-2-ring (modified)
Increasing-ring (modified)
0
1000000
2000000
3000000
4000000
5000000
Energieverbrauch (J)


Variation von BCAST und NB
Zeitverbrauch
8
12
16
24
32
BCAST
Long (modified)
Increasing-2-ring (modified)
Increasing-ring (modified)
0
500
1000
1500
2000
2500
3000
Dauer (s)





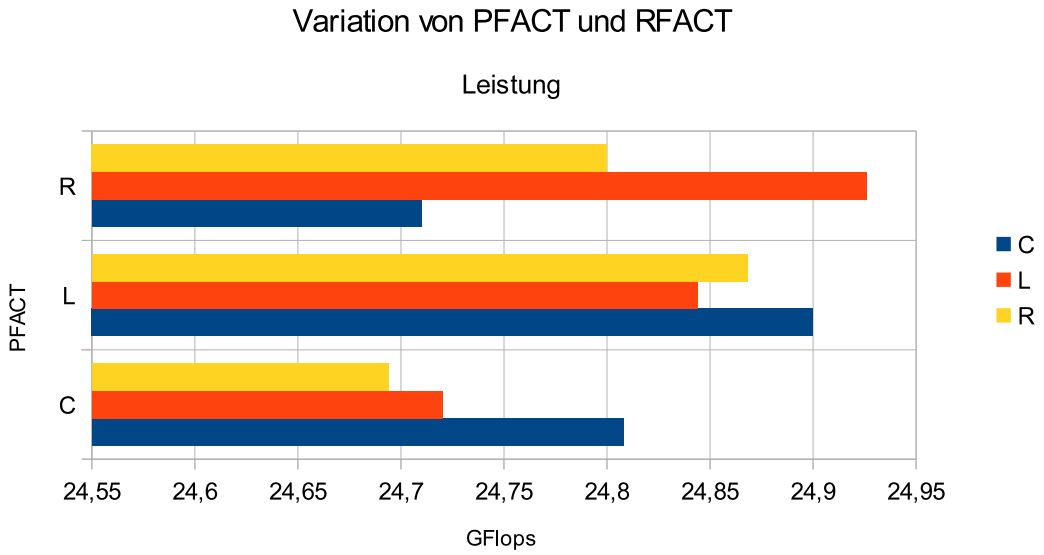
Anmerkung: Die Leistungsaufnahme ist fünfmal höher, die Leistung hingegen 30% geringer als bei einem Cluster der Größe 1. Dadurch ist die Energieeffizienz sehr viel schlechter als bei einem Cluster der Größe 1.

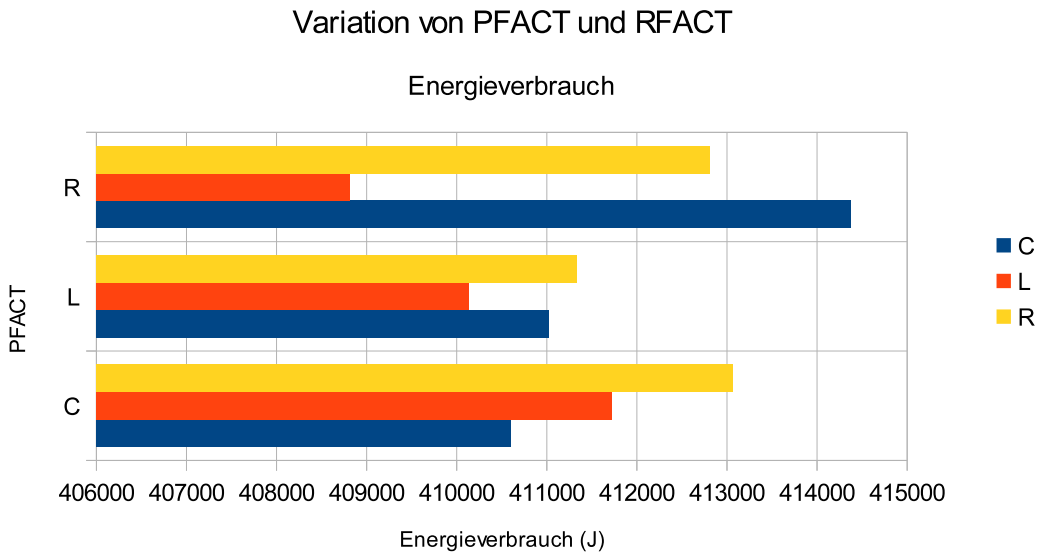
Beobachtung: „Long (modified)“ hat die beste Leistung bei dem geringsten Energieverbrauch. Zu kleine Werte für NB verschlechtern die Leistung und erhöhen den Energieverbrauch. Die Wahl von BCAST verstärkt diese Unterschiede noch.

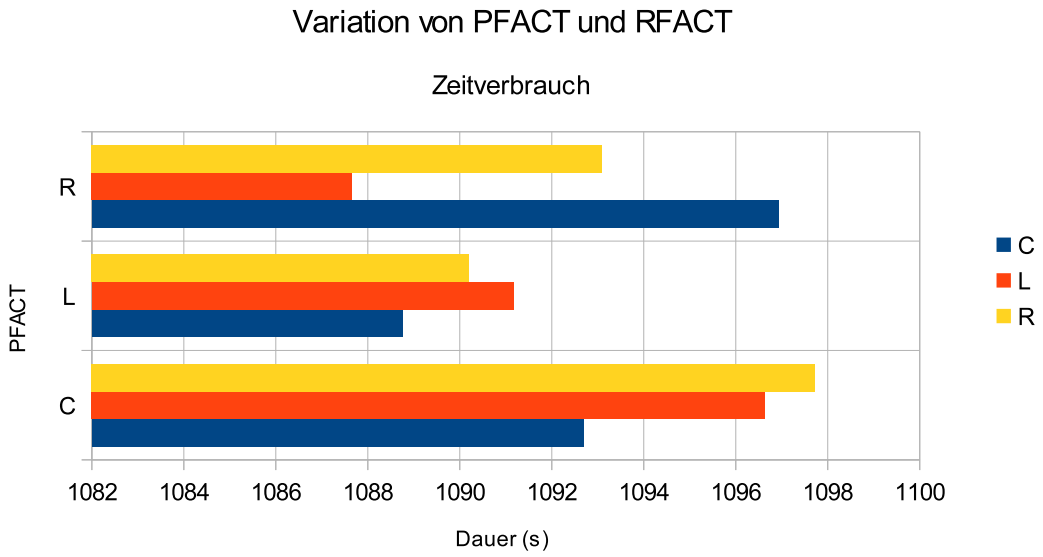
Einschätzung: Eine schlechte Wahl für NB macht bei der Rechenleistung 30-40% und beim Energieverbrauch 30% aus. Ein schlechte Wahl von BCAST wirkt sich mit 25-35% aus.

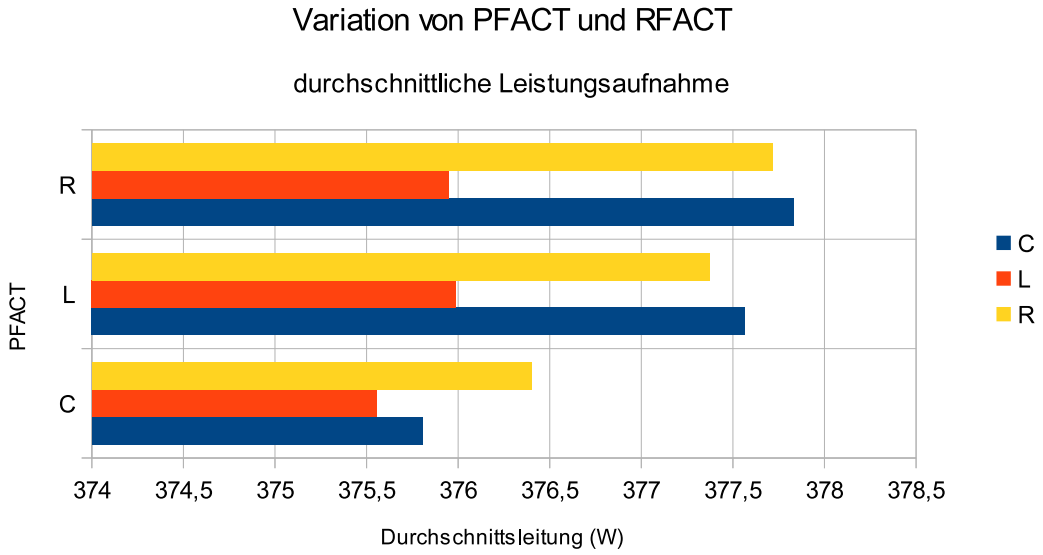
3.2.4 Variation von PFact und RFact

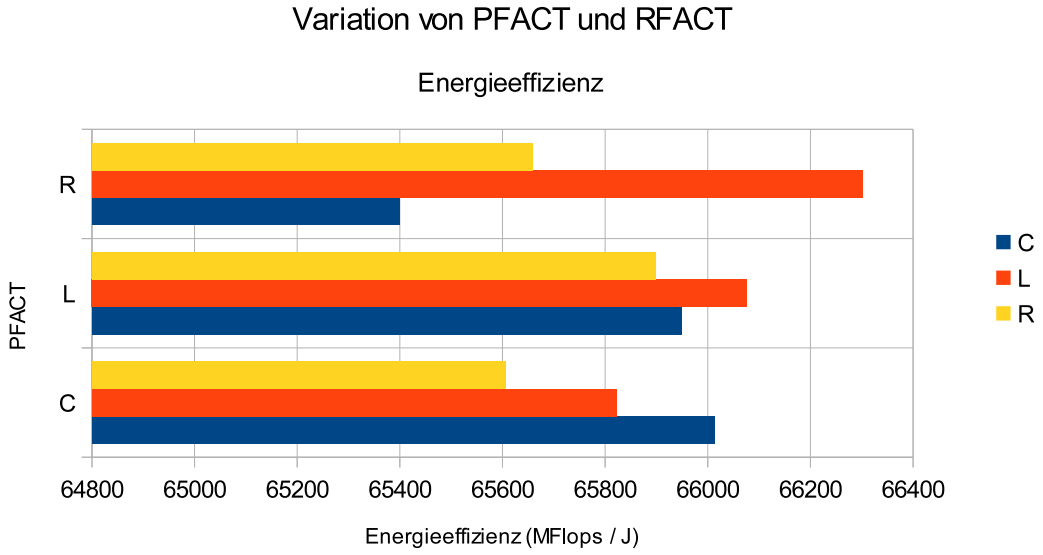
Hier wurden die Parameter PFact und RFact verändert und Messungen für PFact, RFACT = {L,R,C} durchgeführt.






Variation von PFACT und RFACT
Zeitverbrauch
PFACT
R
L
C
1082
1084
1086
1088
1090
1092
1094
1096
1098
1100
Dauer (s)
C
L
R


Variation von PFACT und RFACT
durchschnittliche Leistungsaufnahme
PFACT
R
L
C
374
374,5
375
375,5
376
376,5
377
377,5
378
378,5
Durchschnittsleitung (W)
C
L
R


Variation von PFACT und RFACT
Energieeffizienz
PFACT
R
L
C
64800
65000
65200
65400
65600
65800
66000
66200
66400
Energieeffizienz (MFlops / J)
C
L
R

Beobachtung: Die mit Abstand beste Kombination ist PFACT = R (right-looking panel factorization) und RFACT = L (left-looking recursive factorization). Allerdings sind für RFACT andere Werte als L in Kombination mit PFACT = R vergleichsweise schlecht.

Einschätzung: Diese Unterschiede sind im Vergleich zu anderen Parametern kaum relevant, da die Änderungen sich im Rahmen von weniger als 1% bezüglich Rechenleistung und Energieverbrauch bewegen.

4 Fazit

Es ist erstaunlich, dass ein Cluster der Größe 5 langsamer ist als eines der Größe 1, obwohl die vermeintlich wichtigsten Parameter wie P, Q, NB und BCAST getestet wurden. Das war sehr überraschend. Die Vermutung liegt nahe, dass Parameter nicht getestet wurden, die bei einem Cluster der Größe 1 kaum Auswirkungen hatten. Diese müssten aber für die fünffache Leistung sorgen, was unrealistisch erscheint.

Die Energieaufnahme war vergleichsweise konstant, wodurch niedrigere Rechenleistung meist zu länger laufenden Messungen führte, was wiederum zu einem höheren Energieverbrauch führte. Dadurch waren die schnellsten Parameter auch die energieeffizientesten. Trotzdem wiesen suboptimale Parameter bei vergleichbarer Rechenleistung oft ziemliche Unterschiede im Energieverbrauch auf.

Zusammenfassend wurde Folgendes herausgefunden, nach Relevanz für die Rechenleistung absteigend geordnet:

1. Die richtige Wahl von P und Q ist essentiell. Alle anderen Parameter außer NB verändern die Vor- und Nachteile der Wahl von P und Q nur relativ gering, verglichen mit den Auswirkungen von P und Q selbst. Am besten ist eine Annäherung an das quadratische Ideal.
2. Die richtige Wahl von NB ist ebenfalls wichtig, gerade wenn die Wahl von P und Q suboptimal ausfällt. Hier sind zu kleine Werte besser als zu große, wenn die Abweichung von Ideal relativ gering ist.
3. Bei der Verwendung eines Clusters der Größe 5 spielt die Broadcast-Strategie eine wichtige Rolle, bei einem Cluster der Größe 1 ist sie relativ unbedeutend. Vor allem im Zusammenhang mit der richtigen Wahl von NB können hier Rechenleistung und Energieverbrauch optimiert werden.
4. Die richtige Wahl der Faktorisierungsstrategie ist relativ unbedeutend, die Auswirkungen für Rechenleistung und Energieverbrauch sind relativ klein.
5. Die richtige Wahl von NBMIN und NDIV spielt fast keine Rolle, da die Auswirkungen für Rechenleistung und Energieverbrauch kaum messbar sind.

Natürlich konnten nicht alle Kombinationen aller Parameter getestet werden. Zumindest aber für einen Cluster der Größe 1 wurden die optimalen gefunden und evaluiert. Für einen Cluster der Größe 5 wurden die meisten idealen Parameter nicht gefunden.

Ferner wurde mit der durch den „GCC“ mit Optimierungsstufe 3 kompilierten Version von HPC-LinPack gearbeitet. Hier wären noch Messungen mit spezialisierten Optimierungen denkbar.