Alexander Schnapp
Max Menges

Intro HPC: Blatt 3

04.11.1014

3.1 Moores Law

Nach *Moores Law* verdoppelt sich die Prozessorleistung alle 18 Monate. Für die Rechneleistung R gilt dann nach a Jahren bei momentaner Leistung vom R_{peak}[1]:

$$R = R_{peak} \cdot 2^{\frac{a}{1.5}} \Rightarrow a = 1.5 \cdot \log_2 \frac{R}{R_{peak}}$$

Damit wird ein Exaflop nach *Moores Law* in 6.28 Jahren erreicht.

$$1.5 \cdot \log_2 \frac{1000}{54.9} = 6.28$$

Mit den Werten der TOP500 Liste jeweils aus dem November 2007 und 2011 ergibt sich:

$$a = \frac{4}{\log_2 \frac{R_{2011}}{R_{2007}}} = 0.94$$

Also ein Vedopplung alle 11.3 Monate statt alle 18 Monate wie von Moore vorhergesagt. Damit wir auch ein Exaflop früher erreicht, und zwar in 3.9 Jahren.

3.2 Amdahls Law

a) Mitthilfe einer Verbessurung die durch Ambdahl beschrieben würde, käme man höchstens auf eine Verbesserung um den Faktor 1,6.

b) Verbessert man die Wurzelberechnung um einen Faktor von 10 benötigt man noch $(20/10{+}80 = 82)$ 82% der zuvor benötigten Zeit. Im Falle der Verbesserung der generellen FP Rechnungen kommt man auf $50 + 50/1,6 = 81,25$. Man Braucht also mit etwa 81% knapp weniger Zeit.

c) Es gilt Amdahls Law:

$$Speedup = \frac{1}{(1-P) + P/N}$$

[1] `http://top500.org/lists/2014/06/`

Löst man dieses nach P auf und setzt $N = 128$ und den gewünschten Speedup von 100 ein so kommt man auf

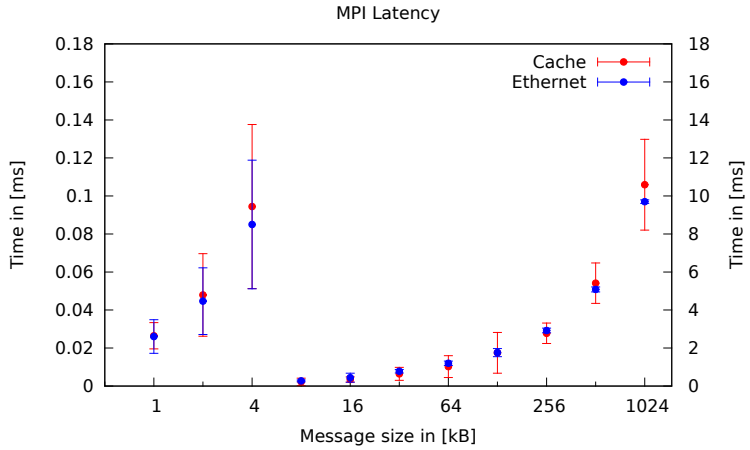
$$P = \frac{\frac{1}{Speedup} - 1}{1/N - 1} = 0.9978$$

und damit auf einen maximalen seriellen Anteil von 0,22%.

3.3 Measure Latency

Der Quelltext liegt unter `../3/3_3/pingpong.cpp`. Makefile zum Compilieren und ausführen liegt bei.

Es werden pro Nachrichtengröße jeweils 8 Nachrichten geschickt (mehr hat MPI irgendwie nicht erlaubt). Das Pingpong Programm wurde zwei mal ausgeführt, einmal mit beiden Prozessen auf einer Maschine („Cache“ im plot) sodass der Nachrichtenaustausch via *Cache* oder Hauptspeicher erfolgt. Beim zweiten Mal Ausführen würden beide Prozesse auf unterschiedlichen Rechnern (`creek04`, `creek06`) ausgeführt. Die Nachrichten müssen also einmal über das Netzwerk („Ethernet“ im plot).



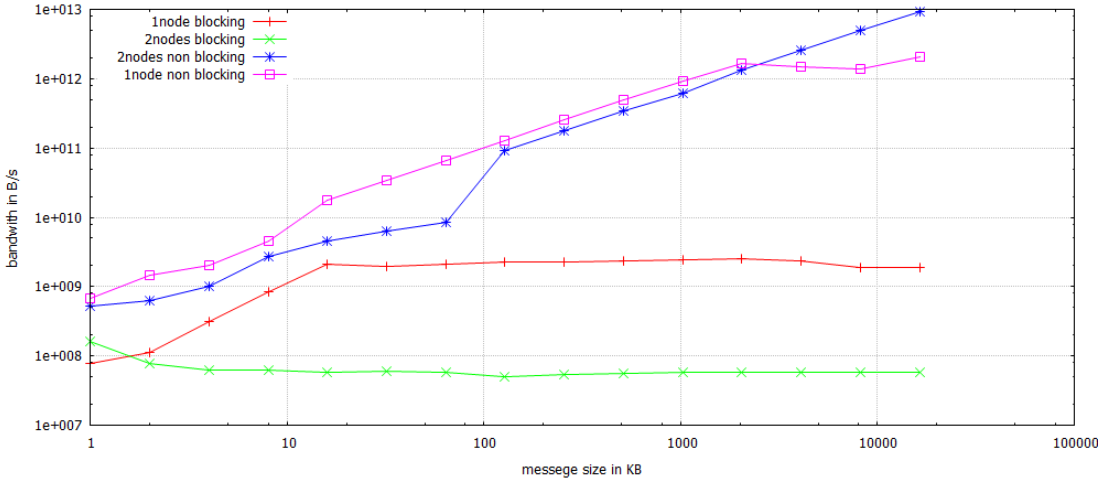
Dabei fallen drei sachen auf:

- Ethernet ist ≈ 10 mal langsamer als der Nachrichtenaustausch auf einem Rechner.
- Die relativen Fehler der Durchschnittslatenz ist bei Ethernet deutlich kleiner. Ethernet ist zwar langsam aber stabil.
- Die Latenz nimmt bei einer nachrichtengröße von $8kB$ deutlich ab. Vermutlich puffert MPI ausgehende Nachrichten und verschickt diese erst wenn der Puffer voll ist (oder nach gewisser Zeit). Das sorgt für hohe Latenz bei kleinen Nachrichten. Da die Latenz bei $8kB$ wieder fällt scheint der Puffer zwischen 4 und 8 kB groß zu sein.

3.4 Measure Bandwidth

Der Quelltext liegt unter `../3/3_4/flood_test.cc`. Makefile zum Compilieren und ausführen liegt bei.

Hier soll die Bandbreite des Systems auf verschiedene Art und weise bestimmt werden. Zunächst senden wir mittels eines blockierenden Sendebefehls anschließend mittels eines nicht blockierenden. Dabei variiert die Größe der gesendeten Daten zwischen 1 KB und 16 MB und verdoppelt sich bei jedem Messpunkt. Des Weiteren wird unterschieden ob die verschiedenen Prozesse auf dem selben oder einem anderen Knoten laufen. Unter Einbezug einer doppelt-logarithmischen Skala ergibt sich folgendes Messergebnis:



Es is zu sehen dass:

- Für ein blockendes send auf einem knoten ist die bandbreite deutlich größer als auf 2 Knoten. Wahrscheinlich da die Bandbreite durch die Ethernet Verbindung begrenzt ist.
- Für das blockende send auf 2 Knoten ist die Bandbreite quasi unabhängig von der Nachrichten Größe wohingegen sie bei einem Knoten erst ansteigt und dann bei etwa 20KB einen stabilen Wert erreicht.
- Bei nicht blockierenden send ist der Unterschied nicht so gravierend. Wahrscheinlich weil für ein nicht blockierendes send , nicht gewartet wird bis die entsprechende Nachricht auch angekommen ist.
- Bei nicht blockierenden send ist die Bandbreite generell höher und erreicht in unseren Messungen stetig an, aus dem selben Grund wie zuvor.