О влиянии регулярных системных прерываний на производительность параллельных программ

Калгин К.В.
Институт вычислительной математики и математической геофизики СО РАН
kalgin@ssd.sscc.ru

Аннотация

В работе исследуется влияние наиболее затратных по ресурсам регулярных системных прерываний (таймер и планировщик). Эти прерывания, в зависимости от аппаратной архитектуры и настроек операционной системы, занимают 0.1-5% времени работы CPU, но могут стать причиной 10-100% ухудшения производительности параллельной программы (например, в массовых операциях [1]). Исследуется влияние этих прерываний на время работы класса параллельных программ с синхронизацией между «соседними» процессами на каждой итерации (например, трафаретные вычисления, синхронный клеточный автомат, явная разностная схема). Приводятся результаты тестирования на вычислительных кластерах. Формулируются меры по уменьшению влияния прерываний на производительность параллельной программы.

1 Введение

Операционная система (ОС) предназначена для решения множества задач, две из которых - обеспечение многозадачности и интерактивности. Решение этих задач обеспечиваетя механизмом прерываний и двумя инструментами – программно-аппаратным *таймером*, отслеживающим реальное время, и программным *планировщиком*, обеспечивающим справедливое выделение квантов времени работы ядер CPU между процессами. В работах [1, 2, 3] показывается, что соответствующие прерывания могут индуцировать 10-100% замедление относительно предсказанного моделью LogP[4] при исполнении коллективных операций на тысячах ядер кластера. При этом, чем больше ядер используется, тем большее замедление наблюдается. Обработка этих прерываний, длительность и частота их возникновения, а также общее замедление параллельной программы зависят как от аппаратной архитектуры, так и от версии ядра ОС Linux.

Кластер	МВС 100К	МВС 10П	НКС-30Т		
Очередь			g7	g6	workq
Intel Xeon	5140	E5-2690	X5670	E5540	E5450
Число ядер в узле	8	16	12	8	8
Процессоров в узле	2				
Ядро Linux	2.6.18	2.6.32	2.6.18		

Таблица 1: Характеристики вычислительных кластеров, на которых проводилось тестирование.

Приведём примеры. До ядра Linux 2.6.21 (в том числе RHEL 4/5, CentOS 4/5, SLES 11) большую роль играют прерывания локального и глобального таймера, запускаемые с частотой 1000 Hz, длительность обработки которых составляет от 1 мкс до 600 мкс в зависимости от архитектуры CPU (см. далее). Начиная с ядра 2.6.21 (RHEL 6, Centos 6, SLES 11 SP1, Clustrx) вместо этого используется High Resolution timer, требующий меньших ресурсов CPU. До ядра 2.6.23 использовался планировщик процессов O(1), его место занял планировщик Completely Fair Scheduler, опять же занимающий меньше ресурсов CPU. Кроме того, от следующих особенностей архитектуры CPU зависит логика планировщика: многоядерность, многопроцессорность, однородный или неоднородный доступ в память.

Именно из-за такого разнообразия возможного поведения, постоянной эволюции аппаратной архитектуры и индивидуальных настроек ОС на различных вычислительных кластерах, в работе не приводится подробный анализ того, почему на том или ином кластере прерывания имеют ту или иную длительность и частоту возникновения.

В работе исследуется влияние прерываний на время работы параллельной программы, имитирующей реализацию на кластере таких алгоритмов как трафаретные вычисления, синхронный клеточный автомат, явная разностная схема. Основная особенность программы - итеративный процесс с синхронизацией между *соседними* процессами на каждой итерации. Приводятся результаты тестирования на кластерах. Формулируются меры по минимизации влияния прерываний на производительность параллельной программы.

Тестирование программ проводилось на кластерах МСЦ РАН МВС-100К, МСЦ РАН МВС-10П, ССКЦ СО РАН НКС-30Т. Характеристики кластеров представлены в Таб. 1.

2 Длительность и частота прерываний

Длительность и частота системных прерываний определялась с помощью следующей последовательной программы. В цикле 10^5 раз запускалась счётная функция, вычисляющая числа Фибоначчи, работающая в среднем T_C

мкс. Длительность исполнения каждого такого запуска сохранялась в памяти, и по завершении цикла записывалась в файл. Анализ полученных времён производился автором (Таб. 1). На Рис. 1 и 2 представлены примеры работы такой программы.

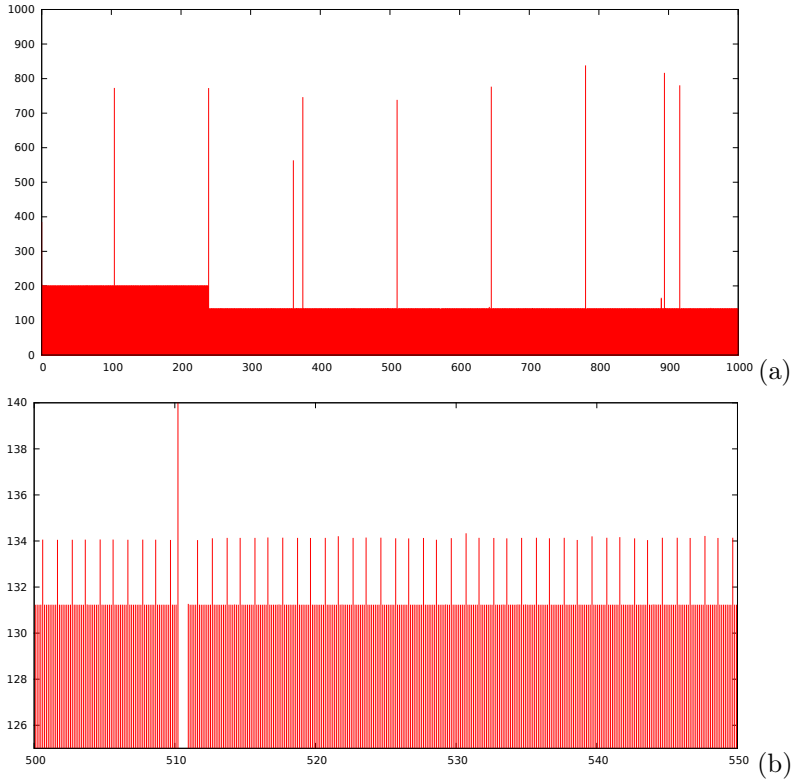

Рис. 1: Длительность каждого из запусков счётной функции за первую секунду работы программы на кластере НКС-30Т, очередь workq. По оси Ox – время в мс от начала работы программы, по оси Oy – время работы счётной функции в мкс. Программа запускалась одновременно на всех ядрах одного узла. Показаны результаты, полученные на 0-м ядре в разных масштабах (a) и (b).

Были замечены следующие явления:

1. Не более первой четверти секунды работы программы время работы счётной функции в 1.5-2 раза больше среднего. Это связано с динамически изменяющейся частотой процессора (Рис. 1).

2. Время работы счётной функции может зависеть от степени загрузки узла – при полной загрузке всех ядер время счёта несколько (<5%)

	Hz	загружено одно ядро		Hz	загружен весь узел	
		ядро 0	ядро 1		ядро 0	ядро 1
НКС-30Т g6/g7	1000	90	13	-	3	1
	8	160	160	-	90	90
МВС 100К	1000	3		-	1	
	8	580		-	580	
МВС 10П	1000	2		100	2	
	40	7		10	7	
	25	13		2	13	
	12	25		0.5	50	

Таблица 2: Длительность (мкс) и частота прерываний на различных кластерах при загрузке одного ядра или всего узла.

меньше, чем при использовании одного ядра. Это тоже связано с динамически меняющейся частотой ядер.

3. На кластере НКС-30Т g6 было замечено, что длительность обработки прерываний зависит от загрузки всего узла (Рис. 2). Прежде всего это относится к 0-му ядру – время обработки прерывания таймера при полной загрузке узла в десятки раз меньше, чем при загрузке только 0-го ядра.

4. Время обработки прерываний на 0-ом ядре обычно больше времени обработки на остальных ядрах.

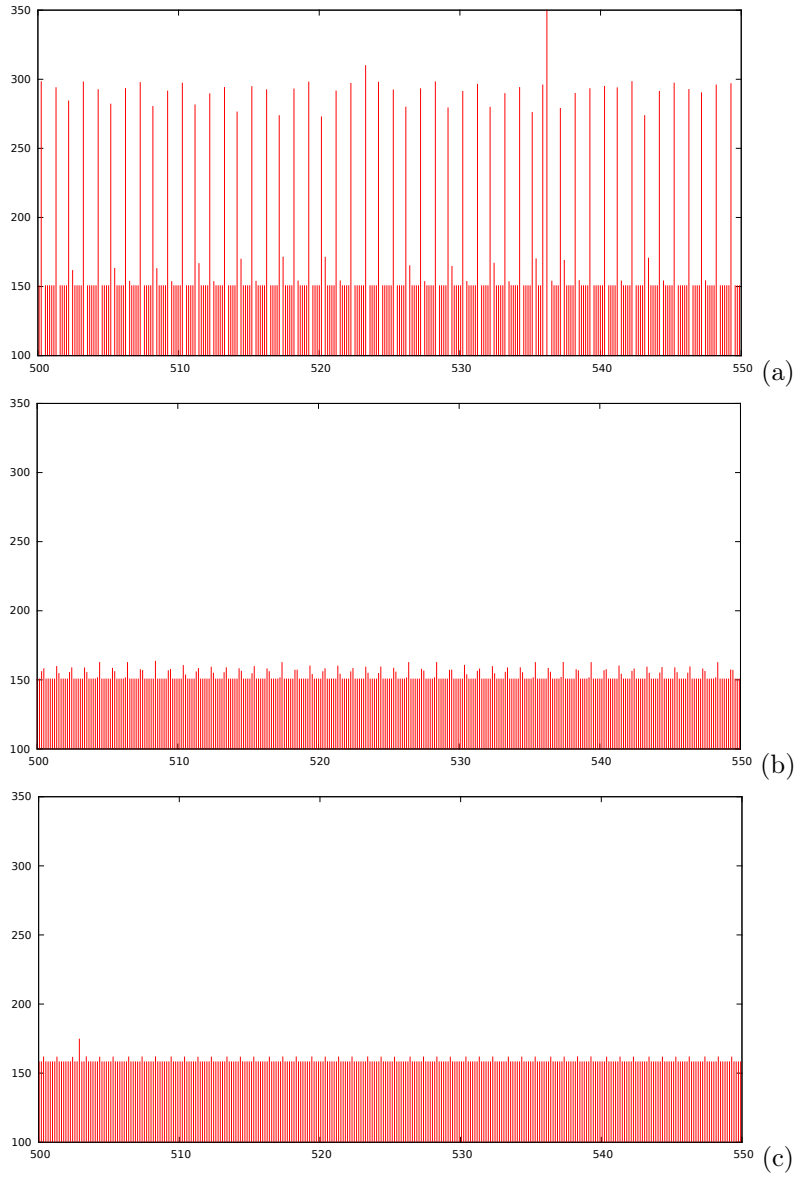

Рис. 2: Длительности работы счётной функции при $T_C = 155$ мкс. Кластер НКС-30Т, очередь g6. По оси Ox – время в мс от начала работы программы, по оси Oy – время работы счётной функции в мкс. Использовалось одно ядро (a,b) или весь узел (c). Результаты с 0-го (a,c) и 1-го (b) ядер.

3 Тестирование на кластерах

Для тестирования использовалась следующая программа, имитирующая параллельную реализацию на кластере таких алгоритмов как трафаретный, синхронный клеточный автомат, явная разностная схема. Пусть запущено P процессов. Каждый процесс выполняет 10^5 итераций. На каждой итерации запускается счётная функция, исполняющаяся в среднем T_C мкс, $1 < T_C < 1000$ мкс. После завершения счётной функции «соседние» по номеру процессы обмениваются массивами данных размера S байт: сообщение пересылается от процесса i к $i+1$ для всех $1 \leq i \leq P-1$, и от процесса i к $i-1$ для всех $2 \leq i \leq P$. Таким образом имитируется один из вариантов реализации параллельного алгоритма, в основе которого лежит деление области на подобласти (domain decomposition) вдоль одной координаты с выделением теневых граней, обновлять которые необходимо каждую итерацию.

Тестирование программы проводилось на кластерах МСЦ РАН МВС-100К, МСЦ РАН МВС-10П, ССКЦ СО РАН НКС-30Т. Характеристики кластеров укзаны в Таб. 1. В качестве коммуникационной библиотеки сообщений использовалась Intel MPI. На кластере запускалась вышеописанная программа на P узлах. На каждом узле порождался один MPI процесс.

Пусть T_P^S – среднее время исполнения одной итерации вышеописанной программы на P узлах при размере сообщения S, тогда T_1 – время работы счётной функции без коммуникаций, $T_1 = T_C$, а

$$N_P^S = T_P^S - T_1$$

– дополнительное время, затрачиваемое на коммуникации между «соседними» процессами.

4 Время исполнения без прерываний

Рассмотрим случай однородных коммуникационных связей между всеми P узлами и отсутствия влияния прерываний на исполнение программы. Время исполнения программы складывается из следующих компонент: времени счёта T_C, времени передачи T^S сообщения размера S в обе стороны между двумя узлами. Тогда время T_p будет равно:

$$T_2^S = T_C + T^S,$$

$$T_P^S = T_C + 2T^S, \text{ при } P \geq 3.$$

Таким образом, при исполнении без прерываний в однородной коммуникационной среде при $P > 3$ должны выполняться равенства

$$T_P^S = T_3^S,$$

$$N_P^S = N_3^S.$$

Тестирование проводилось на кластерах в основном с однородной коммуникационной сетью. Поэтому ненулевую величину $T_P^S - T_3^S$ прежде всего следует отнести к следствиям влияния системных прерываний. В следующем разделе подробно излагается механизм влияния и распространения прерываний в параллельных программах с локальным взаимодействием на каждой итерации.

5 Модель распространения задержек

Опишем модель распространения задержек, порождённых системными прерываниями и распространяющихся через ожидания сообщений передаваемых между «соседними» процессами.

Пусть t_k^i – время, затраченное на исполнение i итераций на узле k, определяется следующим образом:

$$\begin{aligned} t_k^0 &= 0, \quad \forall 1 \le k \le P \\ t_1^{i+1} &= \max\left(t_1^i, t_2^i\right) + T_C + r_1^{i+1} \\ t_k^{i+1} &= \max\left(t_{k-1}^i, t_k^i, t_{k+1}^i\right) + T_C + r_k^{i+1}, \quad 1 < k < p \\ t_p^{i+1} &= \max\left(t_{p-1}^i, t_p^i\right) + T_C + r_p^{i+1} \end{aligned}$$

где r_k^i – случайная неотрицательная величина, реализующая задержки при обработке прерываний, равна

$$r_k^i = f(s_k + t_k^i, s_k + t_k^i + T_C),$$

где s_k – случайная неотрицательная величина, реализующая асинхронность возникновения прерываний на разных узлах за счёт смещения первого прерывания, $f(a,b)$ – функция, вычисляющя время, затраченное на задержки во временном интервале (a,b), обеспечивающая периодичность возникновения прерываний. Например, для кластера НКС-30Т очереди G7 функция f равна:

$$f(a,b) = 160\left(\left\lfloor\frac{b}{125000}\right\rfloor - \left\lfloor\frac{a}{125000}\right\rfloor\right) + 13\left(\left\lfloor\frac{b}{1000}\right\rfloor - \left\lfloor\frac{a}{1000}\right\rfloor\right).$$

Описанный выше процесс является случайным. Математическое ожидание t_p^{1000} – времени вычисления 1000 итераций , M_p, будем вычислять с «точностью в один процент». Это означает, что будет определено математическое ожидание по выборке, M_p^*, в доверительном интервале $(M^* - \frac{M^*}{100}, M^* + \frac{M^*}{100})$ с надёжностью 0.99.

На Рис. 3 представлены значения задержек $T_P^S - T_3^S$, полученные при тестировании, и результаты вычисления $M_p^* - M_3^*$ при соответствующем условиям запуска распределении r. На графике видно, что до 150 узлов модель распространения задержек достаточно точно соответствует результатам тестирования. Начиная со 150 узлов результаты тестирования значительно расходятся с модельными. Автор это объясняет тем, что кластер

МВС-100К на самом деле не является полностью однородным ни по архитектуре узлов, ни по сетвым связям [5]. В частности, во время работы программы на 300 узлах было зафиксировано, что минимальное по узлам затраченное на коммуникации время составляло 43 мкс, а максимальное – 47.5 мкс.

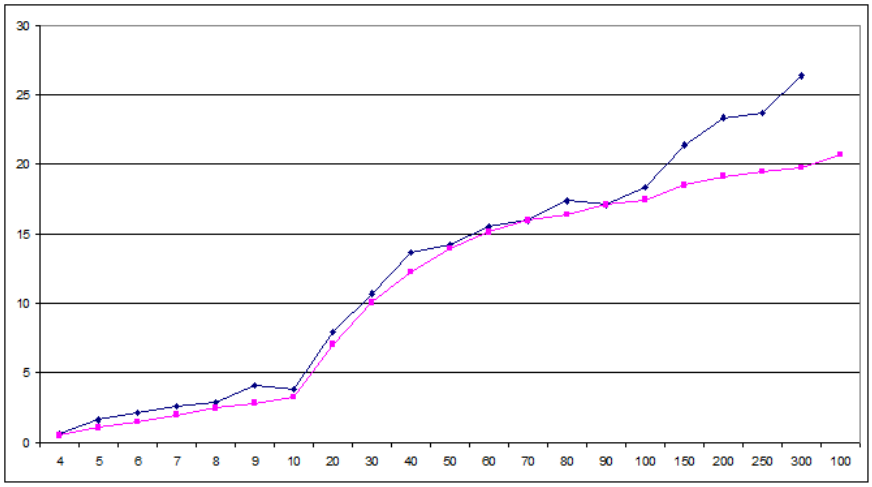

Рис. 3: Синий (верхний) график – значения $T_P^S - T_3^S$, полученные на кластере МВС-100К. Красный (нижний) график – модельные значения $M_p^* - M_3^*$. При $T_C = T_1 = 75$ мкс, $T_3^S = 84.6$ мкс, $T^S = 7$ мкс, $S = 1024$ байт.

\

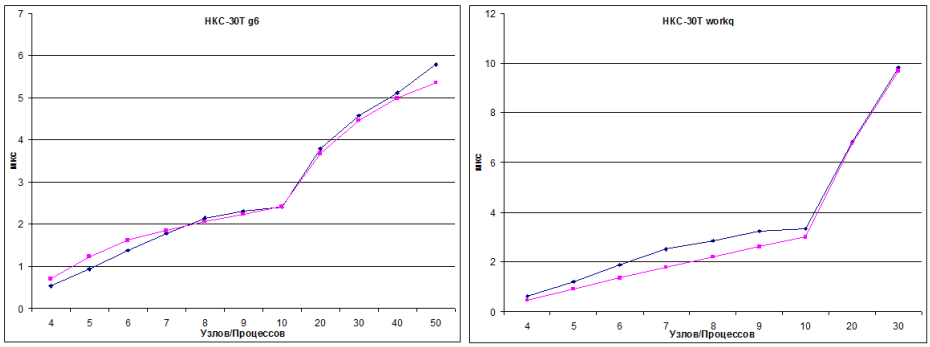

Рис. 4: Синий график – значения $T_P^S - T_3^S$, полученные на кластере НКС-30Т. Красный график – модельные значения $M_p^* - M_3^*$. При $T_C = T_1 = 80$ мкс, $S = 1024$ байт.

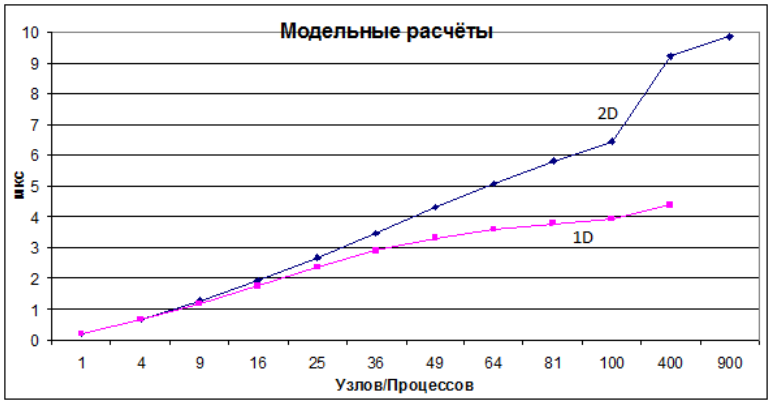

Рис. 5: Модельные значения M_p^*, полученные для одномерной (нижний график) и двумерной (верхний график) декомпозиции области на домены. При $T_C = T_1 = 100$ мкс. Оказывают влияние два прервания – 1000Hz с задержкой 1мкс и 8Hz с задержкой 100мкс.

Заключение

1. Построена модель распространения задержек, связанных с системными прерываниями, в системе параллельно работающих процессов. Показана её адекватность.
2. Показано, что прерывания являются одной из причин деградации производительности параллельного исполнения с ростом числа используемых процессоров.
3. Влияние системных прерываний может быть больше времени коммуникации процессов в разы (Рис. 3).
4. В 2D и в 3D докомпозиции области влияние системных прерываний увеличивается (Рис. 5).
5. Необходимо:
   (a) использовать неблокируемые способы коммуникации, которые будут препятствовать излишней синхронизации процессов;
   (b) внедрять ОС с минимальным «системным шумом»;
   (c) тестирование на определение масштабируемости, производительности, ускорения или эффективности параллельной реализации проводить лишь после четверти секунды работы этих ядер под нагрузкой. Также необходимо учитывать изменения частоты ядер от загруженности узла.

Список литературы

[1] Seetharami Seelam, Liana L. Fong, Asser N. Tantawi, John Lewars, John Divirgilio, Kevin Gildea. Extreme scale computing: Modeling the impact of system noise in multicore clustered systems // International Parallel and Distributed Processing Symposium/International Parallel Processing Symposium - IPDPS(IPPS) , pp. 1-12, 2010

[2] Dan Tsafrir, Yoav Etsion, Dror G. Feitelson, and Scott Kirkpatrick. System noise, OS clock ticks, and fine-grained parallel applications // ICS, page 303-312. ACM, (2005)

[3] Christian Engelmann. INVESTIGATING OPERATING SYSTEM NOISE IN EXTREME-SCALE HIGH-PERFORMANCE COMPUTING SYSTEMS USING SIMULATION

[4] David Culler, Richard Karpy , David Patterson, Abhijit Sahay, Klaus Erik Schauser, Eunice Santos, Ramesh Subramonian, and Thorsten von Eicken. LogP: Towards a Realistic Model of Parallel Computation //PPOPP '93 Proceedings of the fourth ACM SIGPLAN symposium on Principles and practice of parallel programming, p.p. 1-12

[5] http://www.jscc.ru/hard/mvs100k.shtml