Сравнение T-системы и MPI на задаче EP из пакета тестов NPB 2.3

В.А. Евсеенко, М.Н. Иванов, В.Ю. Радыгин

Координаты авторов:

Московский Государственный Индустриальный Университет
evseenko@msiu.ru, ivanov@msiu.ru, radigin@msiu.ru

Введение

В данной работе выполнено сравнение возможностей системы автоматического динамического распараллеливания T-system[1][3] с другим, наиболее широко используемым на данный момент, средством написания распределенных задач MPI. Сравнивались реализация задачи из стандартного набора тестов NASA, написанная на языках t2cp (T-system) и Fortran 77 (MPI). В качестве реализаций для MPI использовались исходные тексты задачи, взятые ровно в том виде, в котором они выложены на сервере NASA [1].

Тестовой задачей был выбран тест «Embrrassingly parallel» (EP). Задача основана на порождении пар псевдослучайных нормально распределенных чисел. Для удобства сравнения результата, по тербованию задачи EP (как это специфицировано NASA), строится гистограма и считаются суммы полученных пар чисел.

Реализация EP на T-системе

Рассмотрим реализацию данной задачи на T-языке. Общий алгоритм описан на сервере NASA [2]. Изучение алгоритма быстро показывает, что задачу можно легко дробить по размеру отрезка для индекса в рекуррентных соотношениях. Расчет каждого отрезка будет абсолютно независим и для отрезка можно получить собственную гистограмму. Таким образом, передача данных между параллельными процессами будет происходить лишь в самом конце, для сложения полученных гистограмм в одну общую. На этой идее основана реализация теста под MPI и на этой же идее основана наша реализация на T-языке.

Структура реализации теста на T-языке изображена на рис. 1 на стр. 2. Задачу распараллеливания в нашей реализации выполняет функция iterator. Она рекурсивно вызывает саму себя, порождая тем самым бинарное дерево вызовов функций губиной 10. Таким образом, мы получаем 1024 гранулы параллелизма. А затем эти же вызванные функции, соответствующие внутренним узлам дерева, собирают все 1024 гистограммы в одну.

Результаты сравнения

В процессе сравнения тест проходил испытания на различных кластерах с различным количеством рабочих процессоров и с различными показателями их тактовой частоты. Но наибольший интерес для нас представляет результат, полученный на кластере НИВЦ МГУ, из-за однородности всех его узлов.

[1]Разработка Института программных систем Российской Академии наук

```
// Определение основных структур
...
// Обычная C функция, которая подсчитывает
// гистограмму для полуотрезка [k, k+packet_len)
void make_gist(u64 k,u64 packet_len, u64 a, u64 s, pack_res *res)
...
// T-функция, дробящая на гранулы
‘func iterator(parm)-> (res);// parm - отрезок, res - гистограмма
   ...
   if(params.rec_count == 0 ){// расчет гистограммы для отрезка parm
      // результат формируется в my_res
     ...
   }else{ // разбиваем отрезок пополам: t1 и t2
     ...
     t1‘newPackedHolder(sizeof(pack_prm));
     t2‘newPackedHolder(sizeof(pack_prm));
     ...
     ‘[res1]=iterator(t1); // рекурсивный вызов для t1
     ‘[res2]=iterator(t2); // рекурсивный вызов для t2
     // суммирование гистограмм
     ...
   }
‘res <== my_res;
‘end_func

‘func tmain(arg) -> (rc);
‘vars res,prm,retval;
   ...
  ‘[res]=iterator(prm);
   ...
‘end_func
```

Рис. 1: Структура T-программы для реализации EP

Структура кластера НИВЦ МГУ. Кластер содержит 13 узлов:

– 12 машин 2×PentiumIII 500

– PentiumIII 500

Основные 12 машин, помимо FastEthernet, связаны при помощи SCI.

Кратко характеристики межпроцессорных связей кластера можно описать следущим образом: латеность для SCI — 5.6 мксек, для Fast Ethernet — 158 мксек (для сообщений нулевой длины), скорость для SCI — 80 MB/sec, для Fast Ethernet — 10 MB/sec (для больших сообщений).

Во время тестирования количество процессоров изменялось от 1 до 25. Для полученных результатов был получено графическое представление отображенное на рис. 2 на стр. 3.

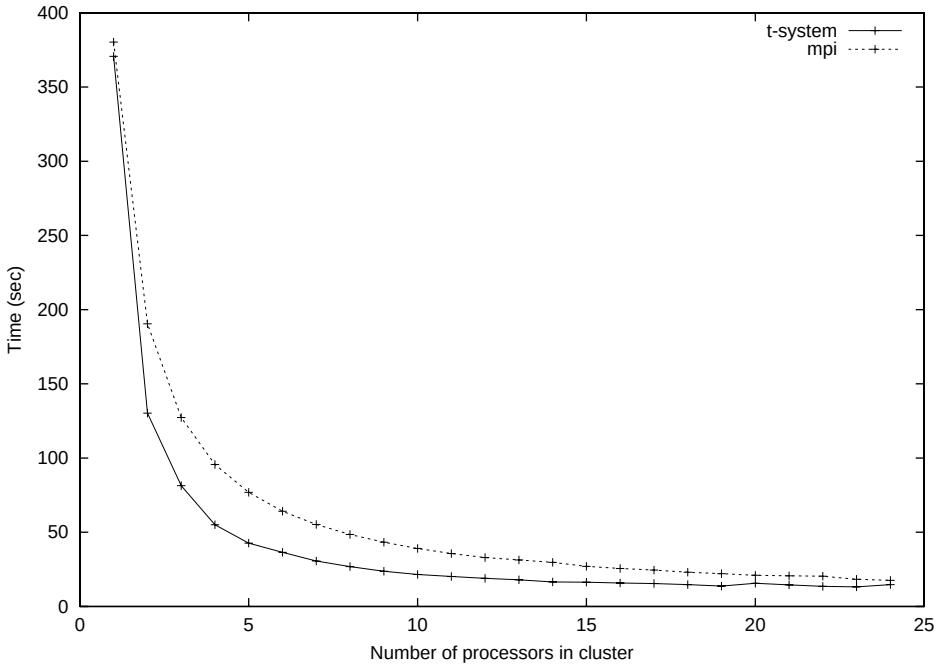

Рис. 2: График зависимости времени вычисления от числа задействованных процессоров

Важно подметить, что T-система использовала в качестве средства связи FastEthernet, а MPI программа на Fortrane 77 использовала SCI. Несмотря на эту заметную разницу, версия на t2cp не только не отстает от MPI версии, но и обгоняет ее практически на любом сочетании процессоров.

Анализ результатов

Для более полного анализа были расчитаны дополнительные параметры:

N — количество процессоров;

t_s — время в секундах;

t_p — время в процентах;

K — коэффициент ускорения;

f — функция утилизации.

Функция утилизации рассчитывается по формуле: $f = \frac{t(1)}{t(n)*n}$, где n — число процессоров, а $t(x)$ — время расчета задачи на x процессоров. Она показывает насколько отличается реальный коэффициент ускорения от "желаемого" линейного роста. Обычно ожидается, что $0 \leqslant f \leqslant 1$. В нашем случае функция утилизации иногда превышает единицу. Для объяснения этого существует много версий. Мы приведем наиболее правдоподобную. Она заключается в том, что из-за локализации данных процессор лучше работает с кэшем, отсюда и получается более чем линейное ускорение. В таблице 1 на стр. 4 приведены полученные результаты для обоих реализаций.

N	t_c T-sys	t_c MPI	t_p T-sys	t_p MPI	K T-sys	K MPI	f T-sys	f MPI	$\frac{K_T}{K_{MPI}}$
1	370.08	380.30	100.0	100.00	1.000	1.00	1.00	1.00	1.00
2	169.97	190.50	45.92	50.09	2.177	2.00	1.09	1.00	1.09
3	90.45	127.30	24.44	33.47	4.091	2.99	1.36	1.00	1.37
4	60.95	95.70	16.46	25.16	6.072	3.97	1.52	0.99	1.53
5	49.58	76.95	13.39	20.23	7.464	4.94	1.49	0.99	1.51
6	40.52	64.15	10.94	16.87	9.133	5.93	1.52	0.99	1.54
7	35.16	55.25	9.503	14.53	10.52	6.88	1.50	0.98	1.53
8	31.09	48.50	8.400	12.75	11.90	7.84	1.49	0.98	1.52
9	26.95	43.28	7.282	11.38	13.73	8.79	1.52	0.98	1.56
10	24.91	39.07	6.730	10.27	14.85	9.73	1.48	0.97	1.53
11	23.23	35.63	6.276	9.37	15.93	10.67	1.45	0.97	1.49
12	21.46	32.93	5.801	8.66	17.23	11.55	1.44	0.96	1.49
13	20.18	31.37	5.452	8.25	18.33	12.12	1.41	0.93	1.51
14	19.28	29.75	5.209	7.82	19.19	12.78	1.37	0.91	1.50
15	18.61	27.05	5.028	7.11	19.88	14.06	1.32	0.94	1.41
16	18.74	25.58	5.063	6.73	19.74	14.87	1.23	0.93	1.33
17	17.93	24.55	4.844	6.46	20.64	15.49	1.21	0.91	1.33
18	18.19	23.12	4.915	6.08	20.34	16.45	1.13	0.91	1.24
19	17.44	22.14	4.712	5.82	21.22	17.18	1.12	0.90	1.24
20	17.24	21.02	4.658	5.53	21.46	18.09	1.07	0.90	1.19
21	16.70	20.67	4.512	5.44	22.16	18.40	1.06	0.88	1.20
22	16.80	20.40	4.542	5.36	22.01	18.64	1.00	0.85	1.18
23	16.34	18.36	4.415	4.83	22.64	20.71	0.98	0.90	1.09
24	15.74	17.61	4.253	4.63	23.51	21.60	0.98	0.90	1.09

Таблица 1:

Выводы

Конечно, сравнение T-системы и MPI всего на одной задаче не позволяет формировать далеко идущие выводы. Однако, для теста EP имеет место следующее:

– T-система позволяет более эффективно распараллелить задачу;

– в случае T-системы функция утилизации f имеет “более хорошие” значения;

– с ростом числа процессоров “падение” f меньше для случая T-системы—то есть, масштабируемость задачи—лучше

При этом, еще раз подчеркнем, что T-система использовала для межпроцессных передач FastEthernet и TCP/IP, а оригинальная версия программы—SCI и MPI.

Планы на будущее.

Авторы надеются в дальнейшем продолжить сравнение T-системы и MPI на других тестах NASA.

Благодарности

Авторы выражают благодарность НИВЦ МГУ за возможность работы на его кластере, особенно хотелось бы поблагодарить Александра Андреева за оказание огромной помощи в проведении работы.

Список литературы

[1] Исходные тексты тестов NASA. `http://www.nas.nasa.gov/Software`

[2] Официальный сервер NASA, документация по тестам.

`http://www.nas.nasa.gov/Software/npb`

[3] Документация по T-системе и T-языку (ИПС РАН).

`http://www.botik.ru/~t-system`