A proposal of metaheuristics to schedule independent tasks in heterogeneous memory-constrained systems

Javier Cuenca[1], Domingo Giménez[1], José-Juan López-Espín[2] and Juan-Pedro Martínez-Gallar[2]

[1]Universidad de Murcia, Spain, javiercm@dtic.um.es, domingo@dif.um.es
[2]Universidad Miguel Hernández de Elche, Spain, jlopez@uhm.es, jp.martinez@uhm.es

The problem

Mapping independent tasks to the processors in a heterogeneous system.
A master-slave scheme is used.

The system has p processors, with:
speeds $s = (s_0, s_1, \ldots, s_{p-1})$,
available memory $m = (m_0, m_1, \ldots, m_{p-1})$,
start-up $t_s = \left(t_{s_{0,0}}, t_{s_{0,1}}, \ldots, t_{s_{p-1,p-1}}\right)$
and word-sending time $t_w = \left(t_{w_{0,0}}, t_{w_{0,1}}, \ldots, t_{w_{p-1,p-1}}\right)$.
The cost of a basic arithmetic operation in each processors: $a = (a_0, a_1, \ldots, a_{p-1})$.

t tasks, with:
volume of data $i = (i_0, i_1, \ldots, i_{t-1})$,
arithmetic cost $c = (c_0, c_1, \ldots, c_{t-1})$,
and the size of the solutions $o = (o_0, o_1, \ldots, o_{t-1})$.

Real problems:
triatomic molecules simulation,
solution of simultaneous equation models...

Different problem configurations

Fixed arithmetic costs and no communications:
From all the mappings which fulfil the memory restrictions, minimize the maximum of the costs of the tasks assigned to the processors multiplied by the processor arithmetic cost:

$$\min_{\{d/\ i_k \le m_{d_k} \forall k=0,1,\ldots,t-1\}} \max_{j=0,1,\ldots,p-1} \left\{ \frac{1}{s_j} \sum_{l=0/\ d_l=j}^{t-1} c_l \right\} \tag{1}$$

A maximum of p^t assignations.

Variable arithmetic costs and no communication:
The costs of basic operations depend on the problem size ($a = (a_0(x), a_1(x), \ldots, a_{p-1}(x))$).
Equation 1 with the cost of a task $a_j\,(i_l)\,c_l$.

Fixed arithmetic and communication costs:
The order in which tasks are assigned to processors must also be determined.
The execution time must be simulated.
Number of configurations $t!p^t$.

Variable arithmetic and communication costs:
The costs are functions of the problem size.
The execution time is simulated.

Problems with slave multiprocessors:
Each task can be assigned to a group of processors which solves the task in parallel, with a memory limit, which could be the sum of the local memories of each processor.
All the mappings of tasks to sets of processors are considered.
Total number of configurations $t!2^{pt}$.

Other possibilities

Dynamic assignation of tasks: it does not get an optimum mapping, due to memory restrictions.

Our approach: use metaheuristics. At running time low mapping time is necessary.

Adaptative metaheuristics: an initial mapping with a metaheuristic which runs only a low number of iterations. While the slaves work on the solution of the tasks, the master improves the initial solution.

Metaheuristics

The methods considered are:
genetic algorithms (GA),
scatter search (SS),
tabu search (TS),
GRASP (GR).

The metaheuristics are analysed by identifying common routines and element representations.

General metaheuristic scheme:

```
Inicialice($S$)
  while not EndCondition($S$)
    $SS$=ObtainSubset($S$)
  if($|SS| > 1$)
    $SS1$=Combine($SS$)
  else
    $SS1 = SS$
  $SS2$=Improve($SS1$)
  $S$=IncludeSolutions($SS2$)
```

For each metaheuristic:
identify how the functions work,
tune the functions and parameters to the different mapping problems.

Initialice: assignes tasks to processors with the probability proportional to the processor speed.
GA: large population.
SS: S with few elements. An improvement method is used.
TS: S with one element.
GR: works by multiple iterations.

ObtainSubset: some of the individuals are selected randomly.

GA: more probability to configurations with better value.

SS: to select all the elements, or to select the best elements to be combined with the worst ones.

TS, GR: not necessary.

Combine: the selected individuals are crossed and $SS1$ is obtained.

GA, SS: i.e. to exchange half of the mappings.

TS, GR: not necessary.

Improve:

GA: mutation operand. To generate a permutation of p elements. To modify the mapping according to the permutation.

SS: A greedy method which works evaluating the fitness value of the elements obtained with the p posible values (with memory constraints) in each component.

TS: some elements in the neighbourhood of the actual element are analysed, excluding those in a list of tabu elements previously analysed.

GR: local search (greedy).

IncludeSolutions: select elements of $SS2$ to be included en S.

GA: the best ones.

SS: the best ones and the most scattered.

TS, GR: the best one.

EndCondition:

GA, SS, TS: that the best fitness value from the individuals in the population does not change over a number of iterations.
GR: multiple iterations.

Preliminary experimental results

Preliminary results for the simplest problem: the size of each task randomly generated between 1000 and 2000; the arithmetic cost is n^3, and the memory requirement n^2; the number of processors coincides with the number of tasks; the costs of basic arithmetic operations between 0.1 and 0.2 $\mu secs$; the memory of each processor is between half the memory needed by the bigest task and one and a half times this memory. The methods have been tested with different values of the parameters and different versions of the basic routines.
Satisfactory results are obtained when:
GA: the population has 80 elements; the maximum number of iterations is 800, and the maximum number of iterations without improving the solution is 80; each pair of elements is combined with half of the components of each parent; the probability of mutation is 1/5.
SS: S has 20 elements; the maximum number of iterations is 400, and the maximum number of iterations without improving the solution is 40; the combination is that in GA; each element is improved with a greedy method, which works by selecting for the processor with highest execution time a task which could be assigned to another processor; the elements with lowest cost function and those more scattered with respect to the best ones (using a 1-norm) are included in the set.
TS: the neighbourhood has 10 elements, obtained by assigning tasks assigned to the processor with most cost to different processors; the maximum number of iterations is 200, and the maximum number of iterations without improving the solution is 20; the tabu search has 10 elements.
GR: the initial set has 20 elements; the number of iterations is 20; the element selected from S is chosen randomly, with more probability for the elements with better objective function; the element is improved with the method used in SS.
In all the cases the elements in S are initially generated randomly assigning the tasks to the processors, with the probability proportional to the processor speed.
The table compares the mapping time and the simulated obtained with each one of the heuristics and those with a backtracking.

	Back		GA		SS		TS		GR	
tasks	map.	simul.	map.	simul.	map.	simul.	map.	simul.	map.	simul.
4	0.025	3132	0.051	3132	0.065	3132	0.010	3132	0.019	3132
8	0.034	4731	0.028	4731	0.132	4731	0.015	4731	0.024	4731
12	0.058	1923	0.021	1923	0.158	1923	0.016	**2256**	0.029	1923
13	0.132	1278	0.055	1278	0.159	1278	0.016	**1376**	0.024	1278
14	0.791	1124	0.081	1124	0.192	1124	0.017	1124	0.027	**1135**

For big systems and using the different heuristics satisfactory mappings are obtained in a reduced time. In the table the mapping and the simulated times for bigger systems are shown.

	GA		SS		TS		GR	
tasks	map.	simul.	map.	simul.	map.	simul.	map.	simul.
25	0.139	1484	0.259	**1450**	0.010	**1450**	0.045	**1450**
50	0.413	1566	0.429	1900	0.015	1757	0.078	**1524**
100	0.592	1903	0.834	1961	0.022	3018	0.158	**1460**
200	0.825	**3452**	1.540	**3452**	0.079	**3452**	0.293	**3452**
400	3.203	**3069**	2.682	3910	0.375	**3069**	0.698	**3069**

Future work

Tune the parameters of the heuristics.
Apply them to the other possible configurations.
Apply to other mapping problems.

Acknowledgments. This work has been partially supported by the Consejería de Educación de la Región de Murcia, Fundación Séneca 02973/PI/05.