UFRGS

Programação Paralela: estado da arte e novos rumos para a exploração de paralelismo multi-nível

Instituto de Informática – UFRGS – 18 de Agosto de 2006

Roteiro da apresentação

- O Grupo de Processamento Paralelo e Distribuído (GPPD)
- O contexto: acabou o computador seqüencial...
- O problema: a programação tradicional continua seqüencial!
  - Quais alternativas?
  - Quais limitações nas alternativas?
  - Quais rumos possíveis nos x próximos anos?

O grupo de pesquisa

- Grupo de Processamento Paralelo e Distribuído (GPPD)
  - Carissimi, Fernando, Geyer, Navaux, Nicolas, Tiaraju
  - 2 recém-doutores, 8-9 doutorandos, 15 (?) mestrandos, alunos de graduação
- Atua em todos os níveis de Processamento de Alto Desempenho
  - Arquitetura
  - Programação
  - Administração de máquinas
  - Aplicações numéricas
  - Etc...

Projetos Institucionais & Colaborações

- Acadêmicos:
  - INPE CPTEC, LAC;
  - UFRJ/COPPE;
  - UFSM, PUCRS, UNISINOS, UNISC, UCS, UCPel, etc...
- Internacional:
  - INRIA (France): **Équipe Associée desde 2005/12**
  - Pittsburgh (USA); Karlsruhe, Berlin (Alemanha);
  - Barcelona (UPC); Porti
- Industrial:
  - DELL;
  - Hewlett-Packard;
  - Itautec.
  - Altus

Vários níveis de paralelismo

Começando no nível continental: a grade computacional...

Vários níveis de paralelismo

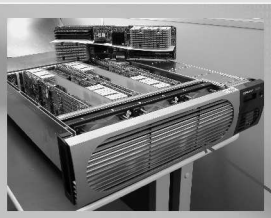
... Passando pelo nível nacional...

... A constelação...

Vários níveis de paralelismo

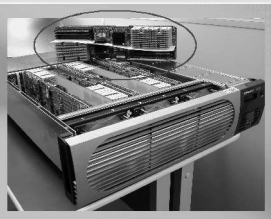
... Em nível local: Máquina dedicada/ Cluster...

Vários níveis de paralelismo



... no nível do chassi com rede de alto desempenho...

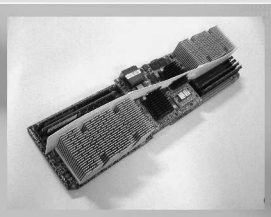
Vários níveis de paralelismo



No nível do nó multi-processado

9

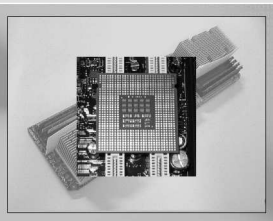
Vários níveis de paralelismo



No nível do nó multi-processado...

10

Vários níveis de paralelismo



... Até o processador multi-core!

11

Qual é teu problema?

- Gerenciamento de tais plataformas é complicado.
  - Acesso, segurança, monitoração, instalação de SW...
  - Depuração de programas!
  - Portabilidade dos programas!
  - Ciclo de vida dos programas!

12

O desafio da programação paralela

- Em todos os níveis, as arquiteturas provêm paralelismo importante.
- Linguagens de programação / algoritmos não são concebidos para aproveitar este paralelismo
  - Algoritmos: o modelo de referência é a máquina de Turing...
  - Modelos paralelos não capturam todos os parâmetros ou são impraticáveis
    - PRAM (memória compartilhada infinita)
    - BSP / CSM (fortemente síncrono)
    - LogP (poucos resultados)

13

Linguagens para programação paralela

- Há muitos problemas não trivialmente paralelos, que exigem muito poder computacional
  - Simulação em climatologia, genoma, petróleo, nuclear,...
- Abordagens para sua programação:
- Fornecer uma abstração de alto-nível, tal como o objeto (Java/Corba)
  - Facilita a migração e o desenvolvimento;
  - Quid da performance?
- Deixar o programador definir vários fluxos de execução:
  - Com comunicação por memória (virtualmente) compartilhada – Threads
  - Com comunicação por rede (Message Passing)
- Fazer com que o compilador resolva...
  - Paralelização automática

14

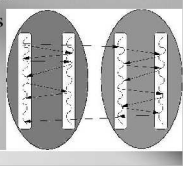
Abordagem 1 - Objetos Distribuídos

- Virtualização do HW/O.S. através de uma Máquina Virtual;
- Encapsulação de dados/métodos no Objeto;
- Publicação / descoberta dos métodos;
- Invocação remota
  - Dados de E/S a serem trocados devem ser serializados!
- Facilita a portabilidade/ubiquidade... Mas com sobrecusto grande!
  - Adaptado para Ubiquous Computing, Pervasive Computing, ...
  - Exemplos: projeto Ibis (H. Bal/Amsterdã); ProActive (INRIA)

15

Abordagem 2 - Processos comunicantes

- Extensões de linguagens seqüenciais clássicas:
  - Bibliotecas de threads
    - Definição, criação e sincronização de fluxos de execução.
  - Troca de mensagens
    - SPMD
    - Bufferização de mensagens
    - Mecanismos bloqueantes ou não



Exemplo de código MPI

Tipo definido pelo MPI

Emissão/recepção de mensagem

```
void  main() {
 int p, r, tag = 103;
 MPI_Status stat;
double valor ;
 MPI_Init(&argc, &argv) ;
 MPI_Comm_rank(MPI_COMM_WORLD, &r);
 if (r==0) {
    printf("Processor 0 sends a message to 1\n");
    valor =3.14 ;
    MPI_Send(&valor, 1, MPI_DOUBLE, 1, tag,
        MPI_COMM_WORLD);  }
 else {
    printf("Processor 1 receives a message from 0\n");
    MPI_Recv(&valor, 1, MPI_DOUBLE, 0, tag, MPI_COMM_WORLD,
   &stat);
    printf("O valor recebido vale %.2lf\n", valor);   }
}
```

17

Abordagem 3 - Paralelização automática

- Paralelização direta e automática: muito **difícil**.
- **Solução**: anotar o código fonte para ajudar o compilador a extrair o paralelismo.
  - Open-MP (estático)
  - Cilk (dinâmico)

- Problema: hoje, só funciona bem com memória compartilhada...

18

Exemplo de código Open-MP

Sequential code	Parallel Open-MP code
double res[10000];	double res[10000];
for (i=0 ; i<10000; i++) compute(res[i]);	#pragma omp parallel for for (i=0 ; i<10000; i++) compute(res[i]);

Conclusão do estado da Arte

- Não existe uma interface única de programação paralela.
- Existem soluções para memória compartilhada e distribuída:
  - MPI / Open-MP
  - Fabricantes incentivam o uso das mesmas
- O que existe não dá conta de:
  - Dinamicidade
  - Heterogeneidade
- A pesquisa está focando esses dois tópicos
  - MPI-2, Cilk, Charm++, Satin/Ibis, Nesl, UPC, etc...
  - A pesquisa (muito) "aplicada" limitou o

Idéias para enfrentar o problema...

- Algoritmos: ir além do "Task Farm"
  - **Divisão & Conquista**, para criação "sob-demanda" do paralelismo.
  - Necessita controle da recursividade/granularidade!
- Linguagem: dar suporte para ambientes distribuídos
  - Parecido ao Open-MP/Cilk, mas com **troca de mensagens**.
- **Compilação on-line** para gerar código eficiente adaptado à máquina alvo
  - Threads vs. Troca de mensagens

Linguagens: suporte ao D&C com mensagens

- Linguagens com novas instruções:
  - Criação de fluxos de execuções (tarefas)
    - Spawn, Fork (como no Cilk)
  - Sincronização / troca de dados
    - Novidade para levar em consideração comunicações em rede, além de memória compartilhada.
      - Semelhante ao "pipe" Unix.
    - Cilk/Open-MP apenas provê acessos concorrentes a dados em memória compartilhada!
    - Levar em consideração a rede

Conclusão

- Plataformas de Grades já existem, ou estão sendo construídas.
  - Apenas para aplicações trivialmente paralelas.
- As linguagens de programação estão evoluindo para acompanhar Grids
  - MPI-2, UPC (Berkeley), ...

- Deve-se prover soluções em novas linguagens para programação eficiente das plataformas da década 2010

Conclusão

- ~~"Aposentem-se antes que complique!"~~

[Luigi Carro 2006]