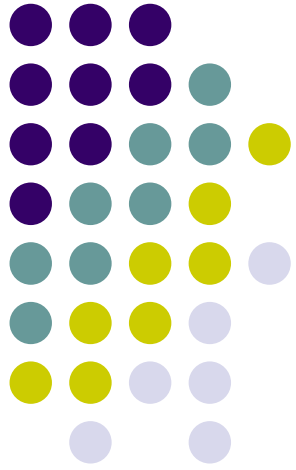
Computação Paralela

ParAspJ – Aplicações Paralelas em Java (parte 1)

João Luís Ferreira Sobral
Departamento do Informática
Universidade do Minho

Novembro 2005



ParAspJ – Aplicações Paralelas em Java

Fases de desenvolvimento de aplicações paralelas

1. Desenvolvimento da aplicação sequencial (cliente – servidor)
2. Acrescentar código de partição do trabalho por vários servidores (cliente - vários servidores)
3. Acrescentar concorrência através da invocação assíncrona de métodos (cliente – Threads – servidor)
4. Distribuir os servidores por várias máquinas

5. **Medir e optimizar o desempenho**

- O ambiente ParAspJ gera automaticamente todo o código necessário para a distribuição dos objectos e distribui os objectos da aplicação pelos nodos de processamento disponíveis

ParAspJ – Aplicações Paralelas em Java

Fases de desenvolvimento para o cálculo de PI

1. **Desenvolvimento da aplicação sequencial (cliente – servidor)**

```
// servidor  calcula o integral entre min e max em intervalos de passo
public class CalcPi {
        public double f( double a ) {
            return (4.0 / (1.0 + a*a));
        }
        public double calc(double min, double max, double passo) {
                double somap=0;
                for(double i=min; i<max; i+=passo)
                        somap += f(i);
                return(passo*somap);
        }
}
```

ParAspJ – Aplicações Paralelas em Java

Fases de desenvolvimento para o cálculo de PI

1. **Desenvolvimento da aplicação sequencial (cont.)**

```
double n=10000000;
System.out.println("Calcular pi com " + n + " intervalos");
double h = 1.0 / n;
CalcPi pc = new CalcPi();
double tapox = pc.calc(0,1,h);
System.out.println("Aproximação  =" + sum + " Dif= " + Math.abs(pi-taprox));
```

ParAspJ – Aplicações Paralelas em Java

Fases de desenvolvimento para o cálculo de PI

2. Acrescentar código de partição do trabalho por vários servidores (cliente - vários servidores)

```
double n=10000000;
System.out.println("Calcular pi com " + n + " intervalos");
double h = 1.0 / n;
CalcPi pc[] = new CalcPi[4];
double aprox[] = new double[4];
for (int p=0; p<4; p++) {
        pc[p] = new CalcPi();
        aprox[p] = pc[p].calc(p/4,(p+1)/4,h);
}
double taprox=0;
for (int p=0; p<4; p++) {
          taprox += aprox[p];
}
System.out.println("Aproximação  =" + sum + " Dif= " + Math.abs(pi-taprox));
```

ParAspJ – Aplicações Paralelas em Java

Fases de desenvolvimento para o cálculo de PI

3. Acrescentar concorrência através da invocação assíncrona de métodos

```
public class PiThread extends Thread {
        double tmin, tmax;
        CalcPi tpic;
        double th;
        double[] tres;
        public PiThread(CalcPi pic, double min, double max, double h, double[] res) {
                tpic = pic; tmin=min; tmax=max; th=h; tres=res;
        }
        public void run() {
                tres[0] = tpic.calc(tmin,tmax,th);
        }
}
```

ParAspJ – Aplicações Paralelas em Java

Fases de desenvolvimento para o cálculo de PI

3. Acrescentar concorrência através da invocação assíncrona de métodos

```
...
CalcPi pc[] = new CalcPi[4];
double aprox[][] = new double[4][1];
PiThread pit[] = new PiThread[4];
for (int p=0; p<4; p++) {
        pc[p] = new CalcPi();
        pit[p] = new  PiThread(pc[p],p/4,(p+1)/4,h,aprox[p][0]);  // aprox[p] = pc.calc(p/4,(p+1)/4,h);
        pit[p].start();
 }
double taprox=0;
for (int p=0; p<4; p++) {
        try {
          pit[p].join();
        } catch(Exception ex) {}
        taprox += aprox[p][0];
}
...
```

ParAspJ – Aplicações Paralelas em Java

Fases de desenvolvimento para o cálculo de PI

4. Distribuir os servidores por várias máquinas

```
public interface ICalcPi extends Remote {
        public double calc(int min, int max, double passo) throws RemoteException;
}

public class PiThread extends Thread {
        ICalcPi tpic;
        …
        public PiThread(ICalcPi pic, int min, int max, double h, double[] res) {
                tpic = pic; tmin=min; tmax=max; th=h; tres=res;
        }
        public void run() {
                try {
                        tres[0] = tpic.calc(tmin,tmax,th);
                 } catch(Exception ex) {}
        }
}
```

ParAspJ – Aplicações Paralelas em Java

Fases de desenvolvimento para o cálculo de PI

4. Distribuir os servidores por várias máquinas

```
...
ICalcPi pc[] = new ICalcPi[4];
double aprox[][] = new double[4][1];
PiThread pit[] = new PiThread[4];
for (int p=0; p<4; p++) {
    try {   // pc[p] = new CalcPi();
        Context ctx = new InitialContext();
        String nome=new String("PS"+(p+1));
        System.out.println("Localizar " + nome);
        pc[p] = (ICalcPi) PortableRemoteObject.narrow(ctx.lookup(nome), ICalcPi.class);
     } catch(Exception ex) {}
     pit[p] = new  PiThread(pc[p],p/4,(p+1)/4,h,aprox[p][0]);  // aprox[p] = pc.calc(p/4,(p+1)/4,h);
     pit[p].start();
  }
...
```

ParAspJ – Aplicações Paralelas em Java

Fases de desenvolvimento para o cálculo de PI

4. Distribuir os servidores por várias máquinas

```
public class CalcPi  extends RemoteObject implements ICalcPi  {
            …
     public double calc(int min, int max, double passo) {
         ….
     }
     public static void main(String args[]) {  // regista o objecto
              try {
                CalcPi pic = new CalcPi();
                PortableRemoteObject.exportObject(pic);
                Context ctx = new InitialContext();
                ctx.rebind(args[0],pic);
                …
              } catch(Exception e) { e.printStackTrace(); }
   }
}
```

ParAspJ – Aplicações Paralelas em Java

Executar a aplicação

- rmic -iiop CalcPi
- tnameserv
- Correr os servidores
  - start java -Djava.naming.factory.initial=com.sun.jndi.cosnaming.CNCtxFactory -Djava.naming.provider.url=iiop://localhost CalcPi PS1
  - start java -Djava.naming.factory.initial=com.sun.jndi.cosnaming.CNCtxFactory -Djava.naming.provider.url=iiop://localhost CalcPi PS2
  - start java -Djava.naming.factory.initial=com.sun.jndi.cosnaming.CNCtxFactory -Djava.naming.provider.url=iiop://localhost CalcPi PS3
  - java -Djava.naming.factory.initial=com.sun.jndi.cosnaming.CNCtxFactory -Djava.naming.provider.url=iiop://localhost CalcPi PS4
- Correr o cliente
  - java -Djava.naming.factory.initial=com.sun.jndi.cosnaming.CNCtxFactory -Djava.naming.provider.url=iiop://localhost Teste