计算科学与工程中的并行编程技术

Parallel Programming Technology in Computational Science and Engineering

都志辉

清华大学计算机系

Email ： duzh@tsinghua.edu.cn

Phone: 62782530

http://hpclab.cs.tsinghua.edu.cn/~duzh

进程组与通信域

都志辉
清华大学计算机系
Email ：duzh@tsinghua.edu.cn
Phone: 62782530

什么是进程组和通信域？

- 进程组：若干进程的集合 group
  - 元素是有顺序的
- 通信域：通信空间（Communication Universe）
  - context的通信域的通信空间进行细分
  - 附加其它的与实现有关的优化信息cache

预定义进程组与通信域

- MPI_GROUP_NULL
  - 无效进程组句柄
- MPI_COMM_NULL
  - 无效通信域句柄

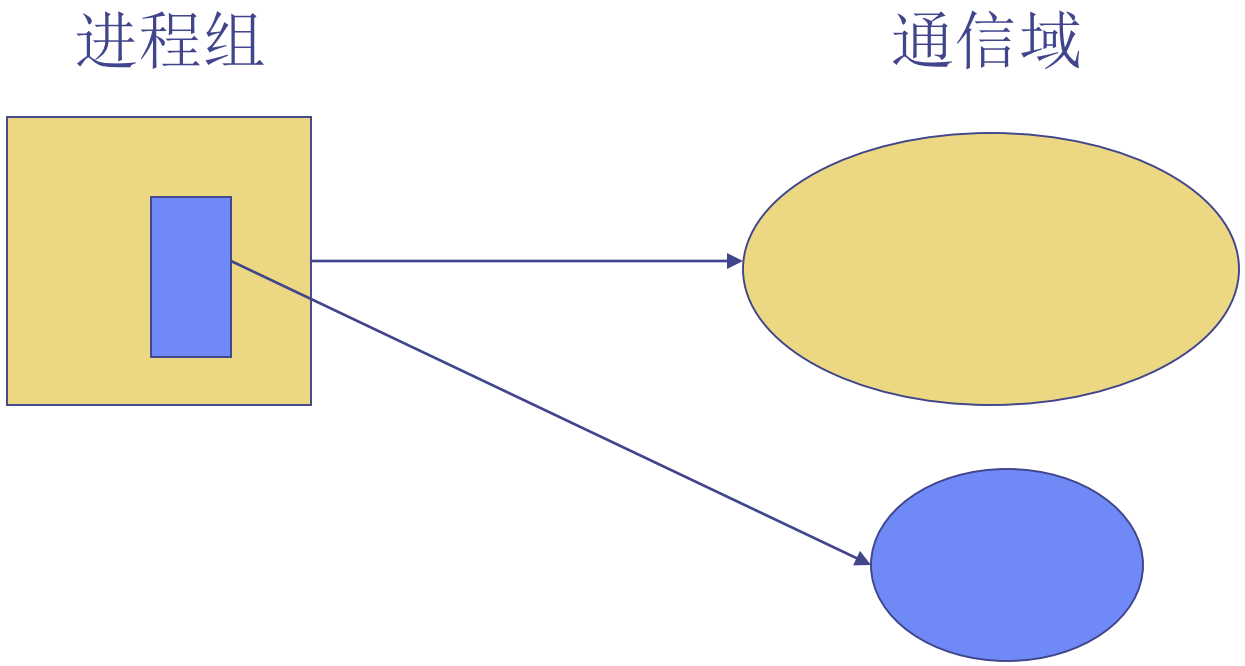
可否作为通信envelop参数？是无效的通信句柄

与

- MPI_GROUP_EMPTY
  - 有效进程组句柄，包括元素个数为0
- MPI_COMM_SELF
  - 有效通信域句柄，包括元素仅为当前进程
- MPI_COMM_WORLD

进程组操作的目的

- 为建立新的通信域服务



建立不同通信域的目的

- 对不同进程，可以有不同的分工

进程组的管理

- 得到通信域对应的进程组

  MPI_COMM_GROUP(comm,group)
- MPI_GROUP_SIZE(group,size)
- MPI_GROUP_RANK(group,rank)

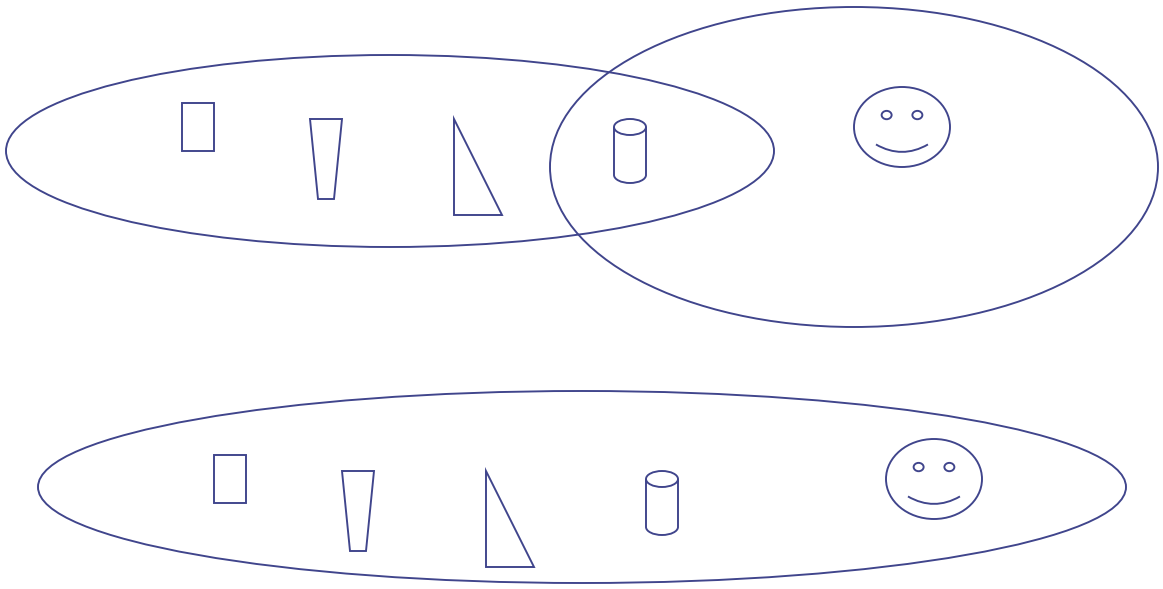
进程组的比较

- MPI_GROUP_COMPARE(g1,g2,result)

MPI_IDENT,MPI_SIMILAR,MPI_UNEQUAL

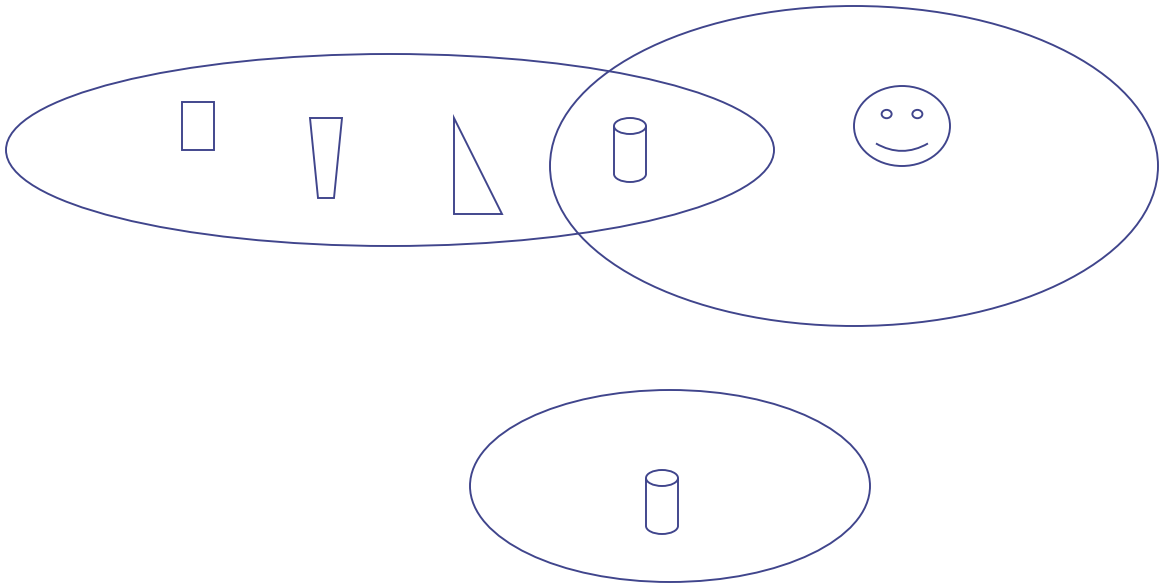
进程组之并

- MPI_GROUP_UNION(g1,g2,newg)



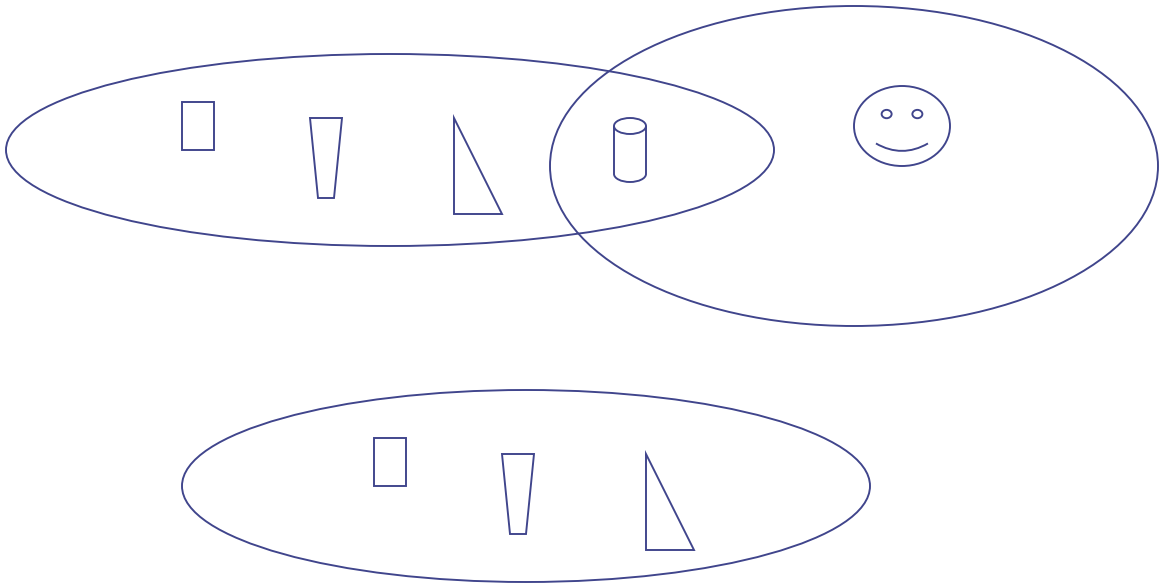
进程组之交

- MPI_GROUP_INTERSECTION(g1,g2,newg)



进程组之差

- MPI_GROUP_DIFFERENCE(g1,g2,newg)

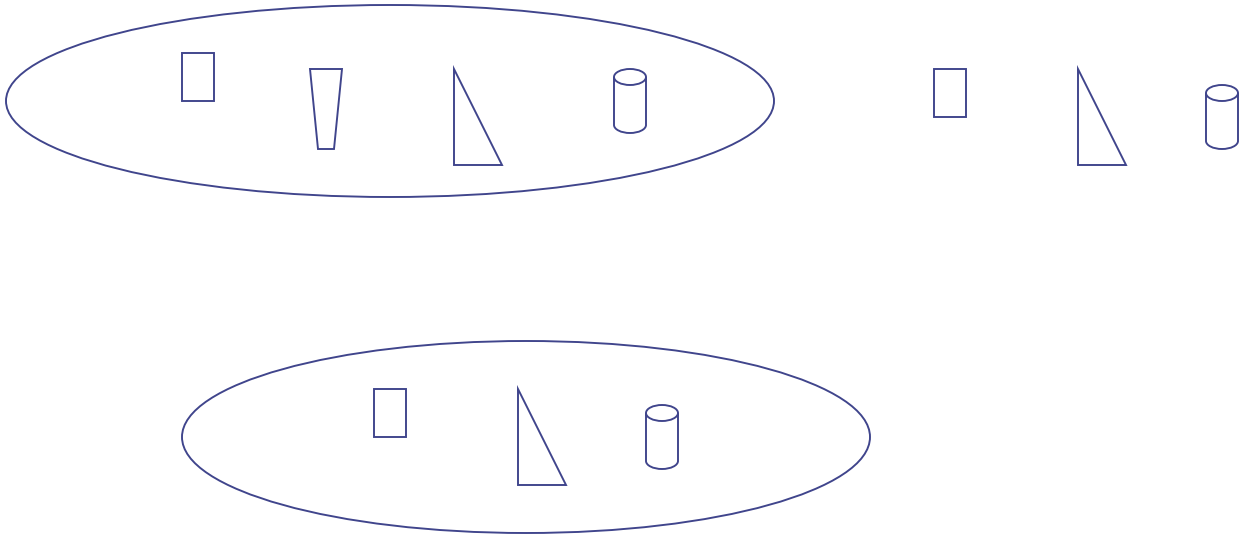


包含在组内

取sub group

- MPI_GROUP_INCL(g,n,ranks,newg)

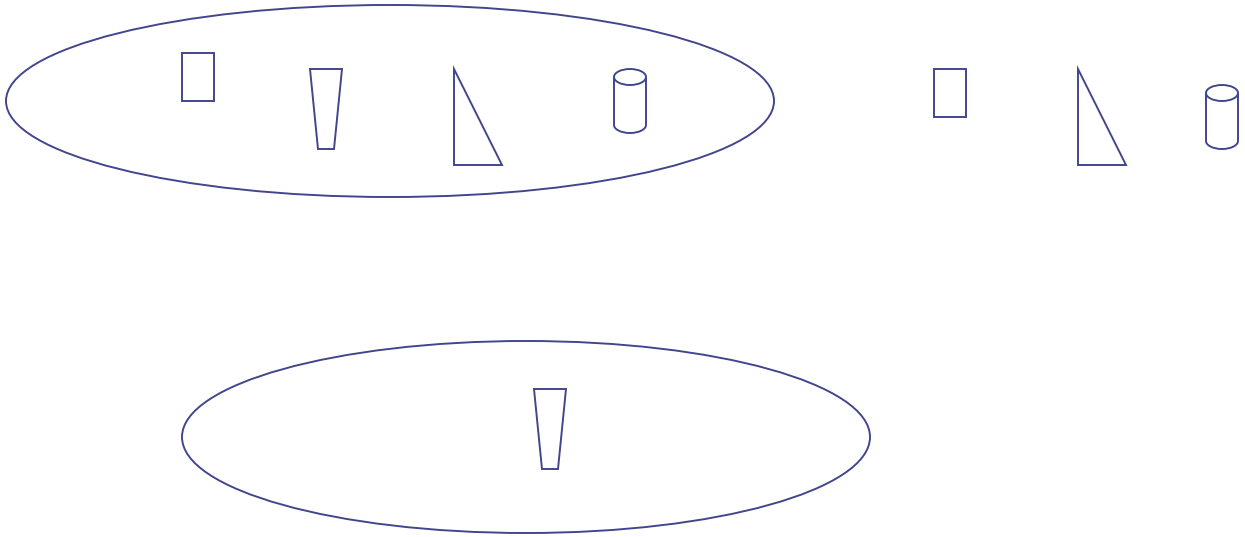
3,{1,2,5}



不包含在组内

排除部分进程

- MPI_GROUP_EXCL(g,n,ranks,newg)



进程组操作的最终目的

- 得到符合需要的进程集合，形成新的通信域

新通信域的产生

- MPI_COMM_CREATE(comm,g,ncomm)

根据进程组产生通信域

例子

这是集合操作吗？

是否需要world中所有进程调用

```
MPI_Comm_group(MPI_COMM_WORLD, &MPI_GROUP_WORLD);

MPI_Group_incl(MPI_GROUP_WORLD, 4, ranks, &subgroup); /* local */
MPI_Group_rank(subgroup, &me);      /* local */

MPI_Comm_create(MPI_COMM_WORLD, subgroup, &the_comm);

if(me != MPI_UNDEFINED)
{
    MPI_Irecv(buff1, count, MPI_DOUBLE, MPI_ANY_SOURCE, TAG_ARBITRARY,
                        the_comm, request);
    MPI_Isend(buff2, count, MPI_DOUBLE, (me+1)%4, TAG_ARBITRARY,
                        the_comm, request+1);
}

for(i = 0; i < SOME_COUNT, i++)
  MPI_Reduce(..., the_comm);
MPI_Waitall(2, request, status);
```

有的应该是null comm

组调用不能在IF语句中

```
IF () {
        组调用
}
```

例子

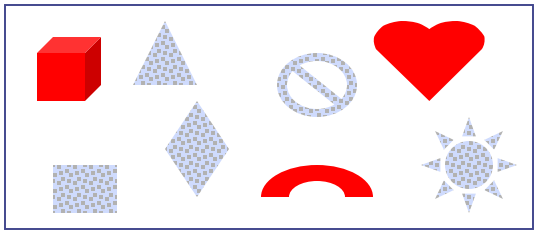
```
main(int argc, char **argv)
{
  int me, count, count2;
  void *send_buf, *recv_buf, *send_buf2, *recv_buf2;
  MPI_Group MPI_GROUP_WORLD, grprem;
  MPI_Comm commslave;
  static int ranks[] = {0};
  ...
  MPI_Init(&argc, &argv);
  MPI_Comm_group(MPI_COMM_WORLD, &MPI_GROUP_WORLD);
  MPI_Comm_rank(MPI_COMM_WORLD, &me);  /* local */

  MPI_Group_excl(MPI_GROUP_WORLD, 1, ranks, &grprem);  /* local */
  MPI_Comm_create(MPI_COMM_WORLD, grprem, &commslave);

  if(me != 0)
  {
    /* compute on slave */
    ...
    MPI_Reduce(send_buf,recv_buff,count, MPI_INT, MPI_SUM, 1, commslave);
    ...
  }
  /* zero falls through immediately to this reduce, others do later... */
  MPI_Reduce(send_buf2, recv_buff2, count2,
             MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);

  MPI_Comm_free(&commslave);
  MPI_Group_free(&MPI_GROUP_WORLD);
  MPI_Group_free(&grprem);
  MPI_Finalize();
}
```

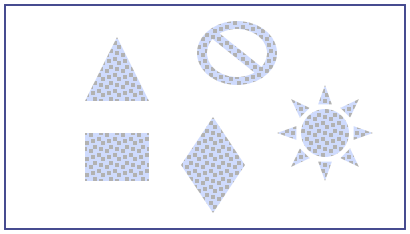
得到多个通信域

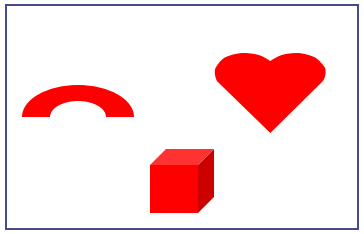
- 通信域的分裂

MPI_COMM_SPLIT(comm,color,key,ncomm)

color决定分组，key决定编号大小







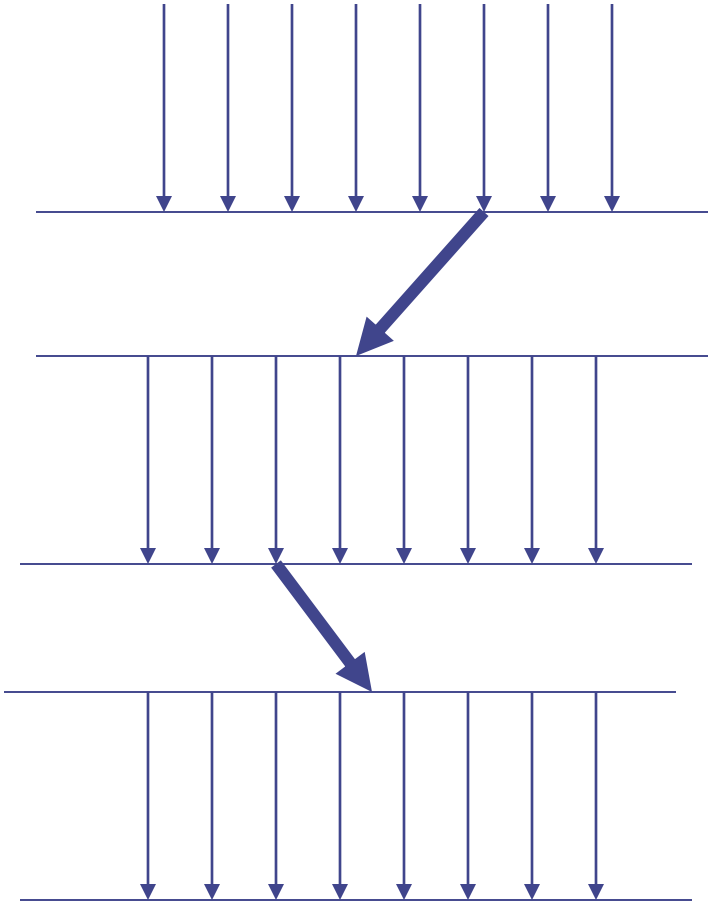
例子

```
Call MPI_COMM_SPLIT(comm, myid %2 ,0,ncomm)
Call MPI_COMM_RANK(ncomm,nmyid)
If (nmyid==0) then
   MPI_COMM_SIZE(ncomm,nsize)
   print  "our group has",nsize, " processes"
End if
Call MPI_BCAST(myid,1,MPI_INTEGER,0,ncomm)
Print * "myid=",myid,"my new id=",nmyid

Mpirun –np 4 t
Mpirun –np 19 t
```

例子：与机械系合作项目
偏微分方程并行迭代求解

- 不同进程选择不同的迭代方向
- 不同进程选择不同的迭代参数

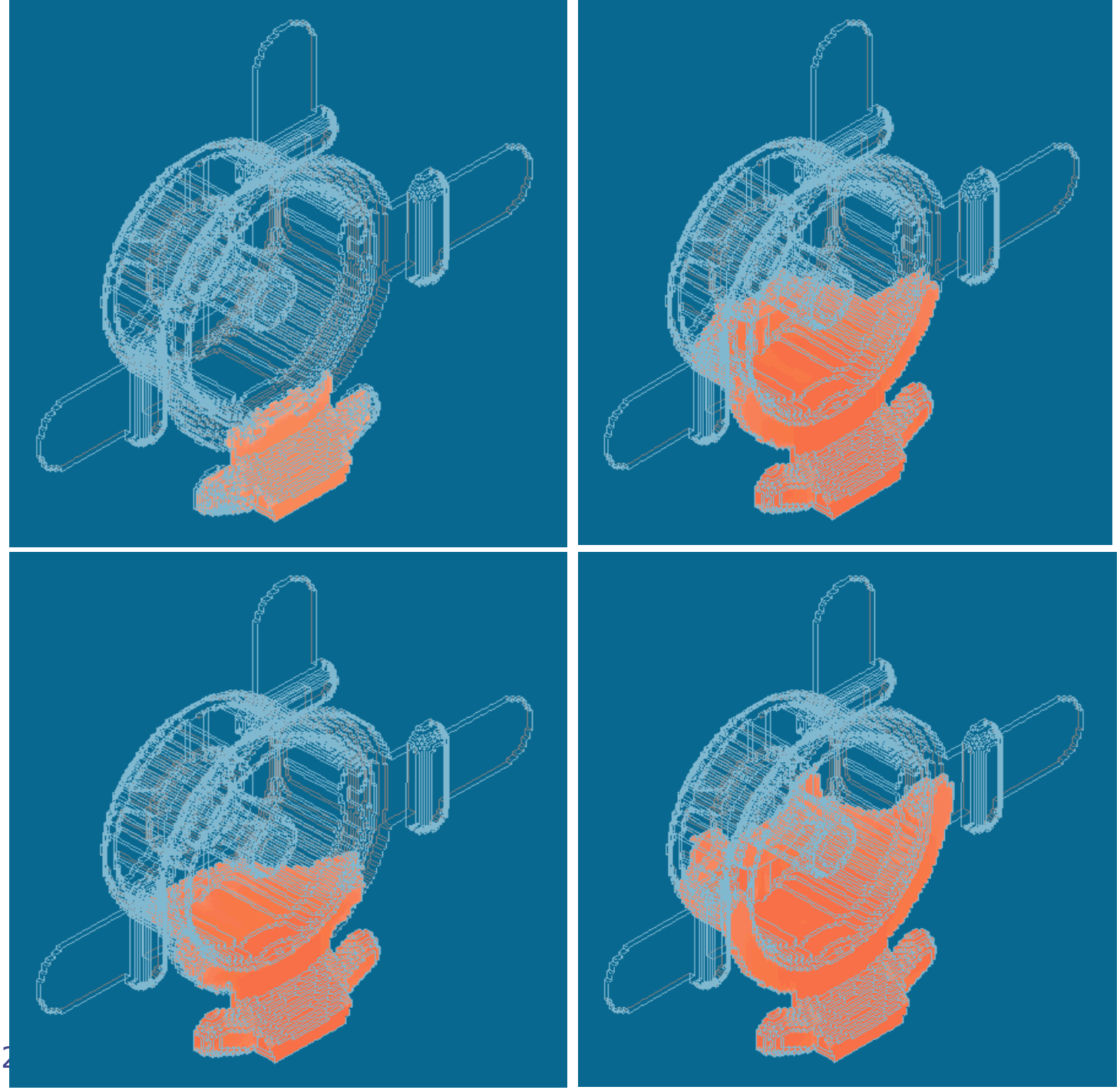
图示



选择收敛速度最快的进程的结果

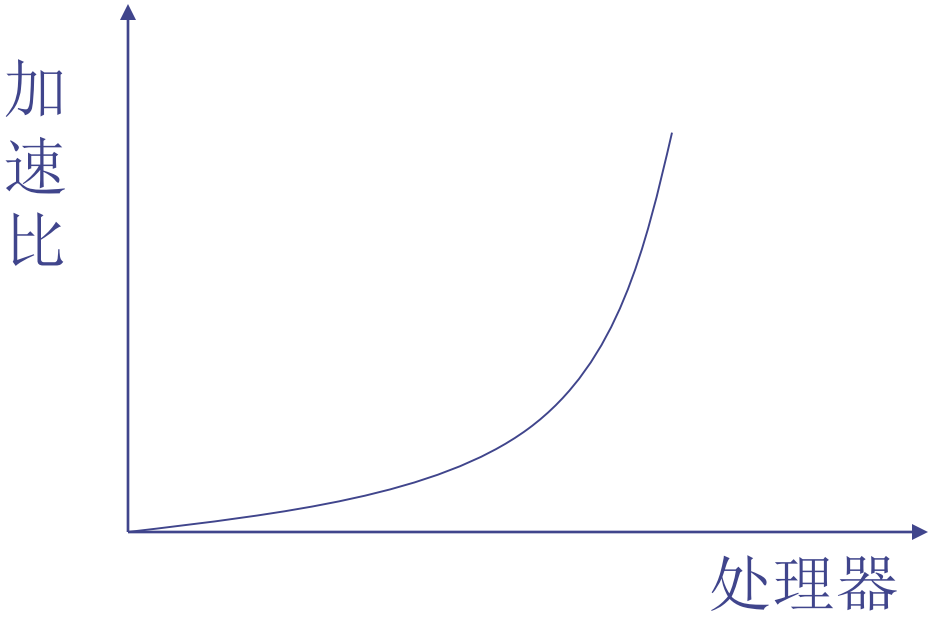
网格剖 分参数

X方向步长（mm）	Y方向步长（mm）	Z方向步长（mm）	铸件单元数	模具单元数	零件体积（cm^3）	零件重量（kg）
1.0	0.8	1.0	46,455	1,302,840	47.41	0.31



效果

- 从一个星期下降到3个小时
- 可能取得超线性的加速比，性能与通信的频率密切相关



超线性加速来自哪里？

综合实例

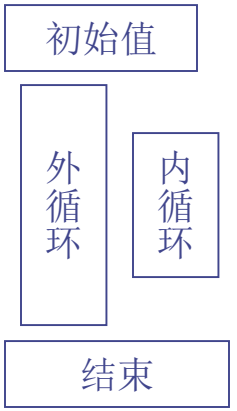
GSAD算法

GSAD简介

- G: Genetic Algorithm（遗传）
- S: Simulated Annealing Algorithm
- D: Downhill Algorithm

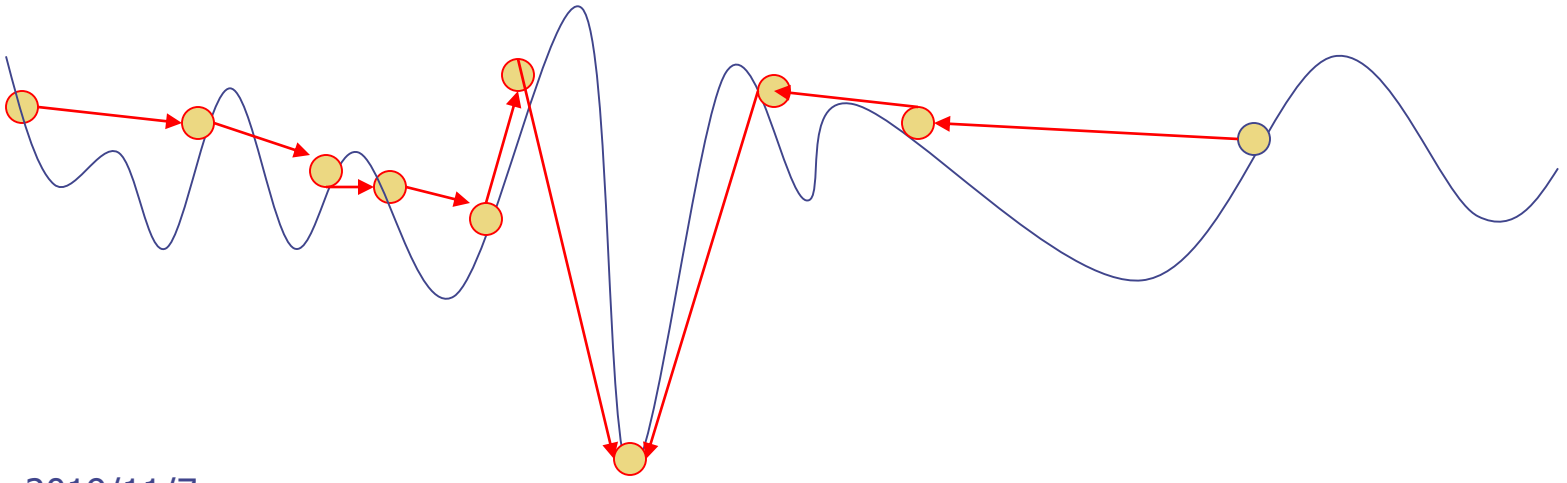
模拟退火算法

- 初始值（初始温度，初始优化点）
- 内循环（特定温度下的马尔可夫链）
- 外循环（是否满足条件，降低温度）



特点

- 关键点：概率接收不好的点，温度缓缓降低
- 优点　收敛性
- 缺点　收敛速度慢，参数敏感



算法

- 温度缓缓降低

  $T=\alpha T,\ 0<\alpha<1$

- 概率接收（差的结果越来越不容易接收）

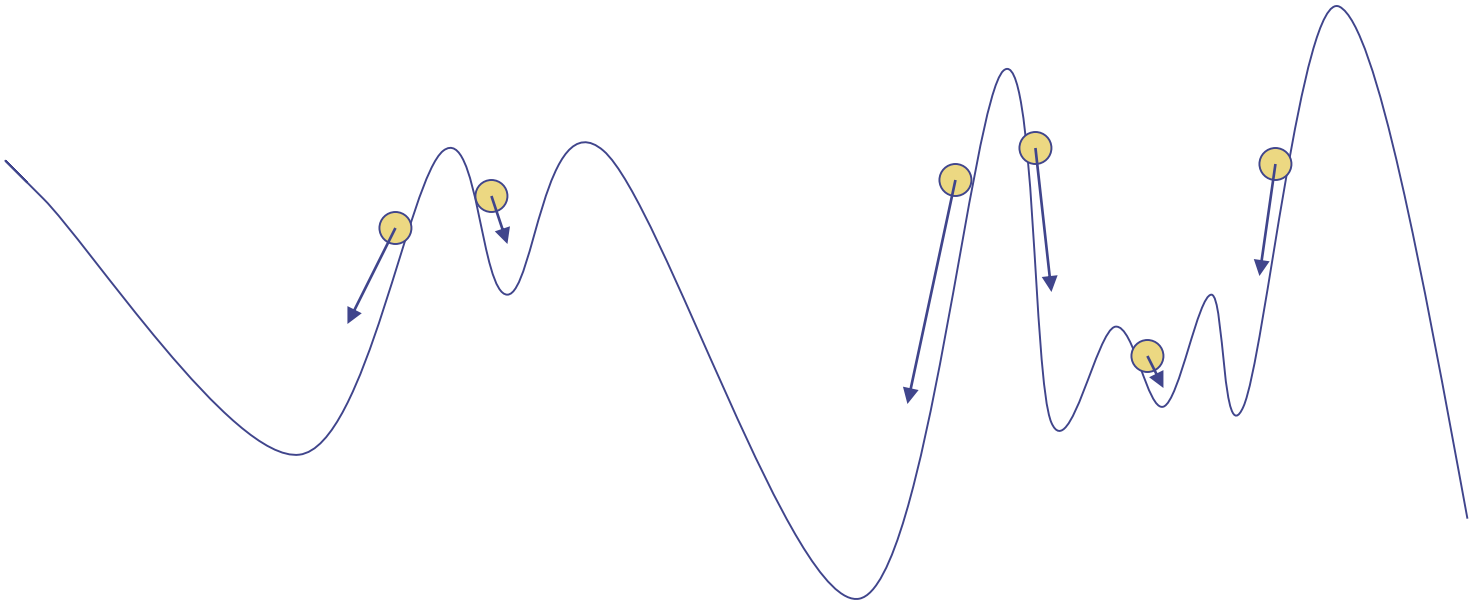
  接收概率= $e^{-\frac{\Delta E}{T}}$

  温度越高越容易接收较差的结果

  评价效果越差越不容易接收

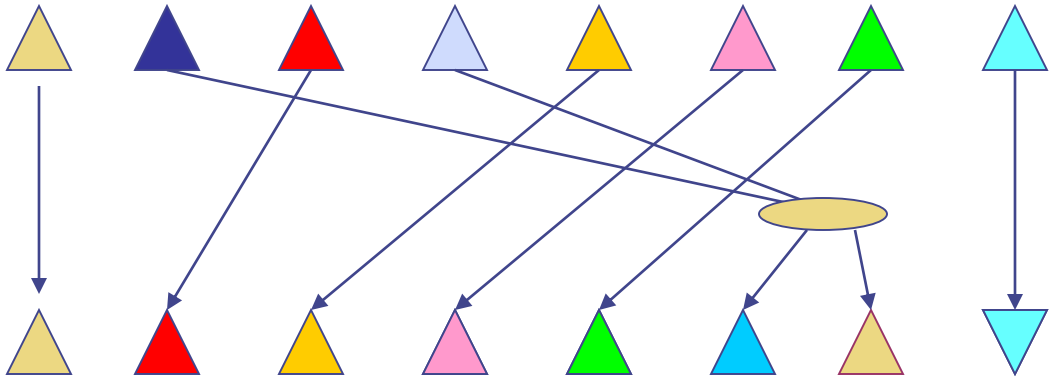
下山法

- 优点　查找速度快
- 缺点　不能保证找到最优值



遗传算法

- 优点 适合并行实现
- 缺点 三个算子如何选择
- 遗传（copy） 交叉（crossover） 变异（mutation）



GSAD算法总结

- GSAD 算法特点
  - 收敛性
  - 收敛速度
  - 扩展性
- 如何实现上述特点
  - 分阶段算法迭加
  - 自适应性
- QCBED 问题

PSO（Particle swarm optimization）算法

群

更新后的值等于局部与全局值综合修正的结果

v[] = v[] + c1 * rand() * (pbest[] - present[]) + c2 * rand() * (gbest[] - present[])

present[] = present[] + v[]

v[]速度，persent[]当前结果，rand ()随机数

c1, c2参数，c1 = c2 = 2. 可以调整

Let S be the number of particles in the swarm, each having a position $\mathbf{x}_i \in \mathbb{R}^n$ in the search-space and a velocity $\mathbf{v}_i \in \mathbb{R}^n$. Let $\mathbf{p}_i$ be the best known position of particle i and let $\mathbf{g}$ be the best known position of the entire swarm. A basic PSO algorithm is then:[11]

```
for each particle i = 1, ..., S do
  Initialize the particle's position with a uniformly distributed random vector: x_i ~ U(b_lo, b_up)
  Initialize the particle's best known position to its initial position: p_i ← x_i
  if f(p_i) < f(g) then
      update the swarm's best known position: g ← p_i
  Initialize the particle's velocity: v_i ~ U(-|b_up-b_lo|, |b_up-b_lo|)
while a termination criterion is not met do:
  for each particle i = 1, ..., S do
    for each dimension d = 1, ..., n do
      Pick random numbers: r_p, r_g ~ U(0,1)
      Update the particle's velocity: v_i,d ← ω v_i,d + φ_p r_p (p_i,d-x_i,d) + φ_g r_g (g_d-x_i,d)
    Update the particle's position: x_i ← x_i + v_i
    if f(x_i) < f(p_i) then
      Update the particle's best known position: p_i ← x_i
      if f(p_i) < f(g) then
        Update the swarm's best known position: g ← p_i
```

练习（使用split）

- 1 进程间两两相互收发
- 2 奇数进程之间和偶数进程之间各自独立地进行循环数据传递，初始值为各自的进程数，各进程都加上自己在MPI_COMM_WORLD中的进程号后传递给下一个进程，各个进程都打印自己的在MPI_COMM_WORLD中的进程号以及接收到的数据。