**并行计算作业**

**并行矩阵乘法算法**

1. 可能存在哪些性能问题？

因为每次都要将矩阵B广播给全部的从进程。需要消耗大量的通信时间；

其次，当矩阵B过大时，需要占用大量内存区域，当系统用调页算法将数据换入pagefile时，需要消耗一段时间来I/O，造成CPU利用率下降，性能降低。

1. 死锁的原因

|  |
| --- |
| pthread\_t\*tids=(pthread\_t\*)malloc(numthread\*sizeof(pthread\_t));  float \*A\_row=(float\*)malloc(N\*sizeof(float));  float \*C\_row=(float\*)malloc(P\*sizeof(float));  struct threadArg \*targs=(struct threadArg\*)malloc(numthread\*sizeof(struct threadArg));  for(i=0;i<numthread;i++){  targs[i].tid=i; /\*编号\*/  targs[i].B=B;  targs[i].A\_row=A\_row;  targs[i].C\_row=C\_row;  } |

在上述代码中，初始化targs数组。每个计算节点有numthread个线程，为每个线程分配相同的B,A\_row,C\_row地址。B矩阵只读，不会发生读写冲突；但是行矩阵A\_row,C\_row需要发生读写，numthread个线程共享同一地址空间，当争抢访问A\_row,C\_row资源时就会发生死锁

另：

|  |
| --- |
| if(myid==0){  ... ...  numsend=j;  for(i=1;i<=M;i++){  sender=(i-1)%(numprocs-1)+1;  MPI\_Recv(C[i-1],P,MPI\_FLOAT,sender,100,MPI\_COMM\_WORLD,&status);  if(numsend<M){  MPI\_Send(A[i-1],N,MPI\_FLOAT,sender,99,MPI\_COMM\_WORLD);  numsend++;  }else{  MPI\_Send(&j,0,MPI\_INT,sender,0,MPI\_COMM\_WORLD);  }  }  ... ...  } |

在0号进程接收发送数据时，当满足numsend<M条件时，

先接收C[i-1]部分数据，再发送A[i-1]部分数据；事实上，从进程计算C[i-1]需要先提供A[i-1]的数据，可能导致实验结果偏慢，因此建议将MPI\_Send放在MPI\_Recv前面；

附录：

实验代码：

|  |
| --- |
| #include"mpi.h"  #include"pthread.h"  #include<stdio.h>  #include<stdlib.h>  #define M 20  #define N 30  #define P 40  /\*创建struct 便于管理信息\*/  struct threadArg{  int tid;  float (\*B)[P]; /\*二维矩阵，用指针数组保存每一行的第一个数的地址\*/  float \*A\_row; /\*矩阵A的一行\*/  float \*C\_row; /\*结果矩阵C的一行\*/  int numthread;  };  void \*worker(void\*arg){  int i,j;  struct threadArg \*myarg=(struct threadArg\*)arg;  /\*平均分配D的所有列\*/  for(i=myarg->tid;i<P;i+=myarg->numthread){  myarg->C\_row[i]=0.0;  for(j=0;j<N;j++){  /\*B的一列与A的一行相乘，存入C对应位置中\*/  myarg->C\_row[i]+=myarg->A\_row[j]\*myarg->B[j][i];  }  /\*printf("C[%d]:%f\n",i,myarg->C\_row[i]);\*/  }  return NULL;  }  int main(int argc,char\*argv[]){  double start\_time=0,end\_time=0;  int i,j;  int myid,numprocs;  MPI\_Status status;  int sender;  float A[M][N],B[N][P],C[M][P];  MPI\_Init(&argc,&argv);  MPI\_Comm\_rank(MPI\_COMM\_WORLD,&myid);  MPI\_Comm\_size(MPI\_COMM\_WORLD,&numprocs);  if(myid==0){  start\_time=MPI\_Wtime();  int i,j;  printf("A:\n");  for(i=0;i<M;i++){  for(j=0;j<N;j++){  A[i][j]=i\*j+1; /\*在0号计算节点，初始化矩阵A\*/  }  }  printf("B:\n");  for(i=0;i<N;i++){  for(j=0;j<P;j++){  B[i][j]=i\*j+1; /\*在0号计算节点，初始化矩阵B\*/  }  }    }  /\*将矩阵B广播给所有其他计算节点\*/  MPI\_Bcast(B[0],N\*P,MPI\_FLOAT,0,MPI\_COMM\_WORLD);  MPI\_Barrier(MPI\_COMM\_WORLD);  if(myid==0){ /\*分配任务，回收结果\*/  int i,j,numsend;  /\*比较进程数量和A的行维数大小，选取二者之中的较小者\*/  j=((numprocs-1)<M?(numprocs-1):M);  for(i=1;i<=j;i++){  /\*将A的每一行分发给从进程\*/  MPI\_Send(A[i-1],N,MPI\_FLOAT,i,99,MPI\_COMM\_WORLD);  }  numsend=j;  for(i=1;i<=M;i++){  sender=(i-1)%(numprocs-1)+1;  if(numsend<M){/\*若数组A还没有发送完数据，则继续发送\*/  MPI\_Send(A[i-1],N,MPI\_FLOAT,sender,99,MPI\_COMM\_WORLD);  numsend++;  }else{  /\*终止sender进程的运行\*/  MPI\_Send(&j,0,MPI\_INT,sender,0,MPI\_COMM\_WORLD);  }  /\*回收来自从进程sender的计算结果\*/  MPI\_Recv(C[i-1],P,MPI\_FLOAT,sender,100,MPI\_COMM\_WORLD,&status);      }  /\*打印结果\*/  for(i=0;i<M;i++){  printf("%d: ",i);  for(j=0;j<P;j++){  /\*printf("%.0f ",C[i][j]);\*/  }  printf("\n");  }  printf("I put an flag\n");  end\_time=MPI\_Wtime();  printf("Past time:%f\n",end\_time-start\_time);  }else{  int i,j;  int numthread=get\_nprocs();/\*获取当前计算节点的CPU数量\*/    printf("cpu number:%d.\n",numthread);/\*get cpu number\*/  /\*创建一组线程，保证每个CPU上运行一个线程\*/  pthread\_t \*tids=(pthread\_t\*)malloc(numthread\*sizeof(pthread\_t));  /\*A矩阵的一行\*/  float \*A\_row=(float\*)malloc(N\*sizeof(float));  /\*C矩阵的一行\*/  float \*C\_row=(float\*)malloc(P\*sizeof(float));  /\*用于统一管理每个线程的A，B，C矩阵信息\*/  struct threadArg \*targs=(struct threadArg\*)malloc(numthread\*sizeof(struct threadArg));  for(i=0;i<numthread;i++){  targs[i].tid=i; /\*编号\*/  targs[i].B=B;  targs[i].A\_row=A\_row;  targs[i].C\_row=C\_row;  targs[i].numthread=numthread;  }    while(1){  int i,j;  /\*接收主进程发来的一行,MPI\_ANY\_TAG表示任意标记的数据都要接收\*/  MPI\_Recv(A\_row,N,MPI\_FLOAT,0,MPI\_ANY\_TAG,MPI\_COMM\_WORLD,&status);  printf("I am thread:%d,debugging: ",myid);  /\*for(i=0;i<N;i++)  printf("A\_row[%d]: %.0f \n",i,A\_row[i]);\*/  /\*检查接受到的tag标记，若为0，则退出\*/  if(status.MPI\_TAG==0) break;  for(i=0;i<numthread;i++){  /\*创建一组线程  第一个参数为指向线程ID。  第二个参数用来设置线程属性。  第三个参数是线程运行函数的起始地址。  最后一个参数是运行函数的参数\*/  pthread\_create(&tids[i],NULL,worker,&targs[i]);  }  for(i=0;i<numthread;i++){  /\*以阻塞的方式等待thread指定的线程结束,等待全部线程计算完成\*/  pthread\_join(tids[i],NULL);  }  /\*从进程返回计算结果给0进程\*/  MPI\_Send(C\_row,P,MPI\_FLOAT,0,100,MPI\_COMM\_WORLD);  /\*printf("I put an flag\n");  for(j=0;j<P;j++){  printf("%d ",(int)C\_row[j]);  }  printf("I put an flag again\n");\*/  }  }  MPI\_Finalize();  } |

试验结果：

|  |
| --- |
| A:  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30  1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59  1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88  1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97 101 105 109 113 117  1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 121 126 131 136 141 146  1 7 13 19 25 31 37 43 49 55 61 67 73 79 85 91 97 103 109 115 121 127 133 139 145 151 157 163 169 175  1 8 15 22 29 36 43 50 57 64 71 78 85 92 99 106 113 120 127 134 141 148 155 162 169 176 183 190 197 204  1 9 17 25 33 41 49 57 65 73 81 89 97 105 113 121 129 137 145 153 161 169 177 185 193 201 209 217 225 233  1 10 19 28 37 46 55 64 73 82 91 100 109 118 127 136 145 154 163 172 181 190 199 208 217 226 235 244 253 262  1 11 21 31 41 51 61 71 81 91 101 111 121 131 141 151 161 171 181 191 201 211 221 231 241 251 261 271 281 291  1 12 23 34 45 56 67 78 89 100 111 122 133 144 155 166 177 188 199 210 221 232 243 254 265 276 287 298 309 320  1 13 25 37 49 61 73 85 97 109 121 133 145 157 169 181 193 205 217 229 241 253 265 277 289 301 313 325 337 349  1 14 27 40 53 66 79 92 105 118 131 144 157 170 183 196 209 222 235 248 261 274 287 300 313 326 339 352 365 378  1 15 29 43 57 71 85 99 113 127 141 155 169 183 197 211 225 239 253 267 281 295 309 323 337 351 365 379 393 407  1 16 31 46 61 76 91 106 121 136 151 166 181 196 211 226 241 256 271 286 301 316 331 346 361 376 391 406 421 436  1 17 33 49 65 81 97 113 129 145 161 177 193 209 225 241 257 273 289 305 321 337 353 369 385 401 417 433 449 465  1 18 35 52 69 86 103 120 137 154 171 188 205 222 239 256 273 290 307 324 341 358 375 392 409 426 443 460 477 494  1 19 37 55 73 91 109 127 145 163 181 199 217 235 253 271 289 307 325 343 361 379 397 415 433 451 469 487 505 523  1 20 39 58 77 96 115 134 153 172 191 210 229 248 267 286 305 324 343 362 381 400 419 438 457 476 495 514 533 552  B:  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40  1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79  1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100 103 106 109 112 115 118  1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97 101 105 109 113 117 121 125 129 133 137 141 145 149 153 157  1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 121 126 131 136 141 146 151 156 161 166 171 176 181 186 191 196  1 7 13 19 25 31 37 43 49 55 61 67 73 79 85 91 97 103 109 115 121 127 133 139 145 151 157 163 169 175 181 187 193 199 205 211 217 223 229 235  1 8 15 22 29 36 43 50 57 64 71 78 85 92 99 106 113 120 127 134 141 148 155 162 169 176 183 190 197 204 211 218 225 232 239 246 253 260 267 274  1 9 17 25 33 41 49 57 65 73 81 89 97 105 113 121 129 137 145 153 161 169 177 185 193 201 209 217 225 233 241 249 257 265 273 281 289 297 305 313  1 10 19 28 37 46 55 64 73 82 91 100 109 118 127 136 145 154 163 172 181 190 199 208 217 226 235 244 253 262 271 280 289 298 307 316 325 334 343 352  1 11 21 31 41 51 61 71 81 91 101 111 121 131 141 151 161 171 181 191 201 211 221 231 241 251 261 271 281 291 301 311 321 331 341 351 361 371 381 391  1 12 23 34 45 56 67 78 89 100 111 122 133 144 155 166 177 188 199 210 221 232 243 254 265 276 287 298 309 320 331 342 353 364 375 386 397 408 419 430  1 13 25 37 49 61 73 85 97 109 121 133 145 157 169 181 193 205 217 229 241 253 265 277 289 301 313 325 337 349 361 373 385 397 409 421 433 445 457 469  1 14 27 40 53 66 79 92 105 118 131 144 157 170 183 196 209 222 235 248 261 274 287 300 313 326 339 352 365 378 391 404 417 430 443 456 469 482 495 508  1 15 29 43 57 71 85 99 113 127 141 155 169 183 197 211 225 239 253 267 281 295 309 323 337 351 365 379 393 407 421 435 449 463 477 491 505 519 533 547  1 16 31 46 61 76 91 106 121 136 151 166 181 196 211 226 241 256 271 286 301 316 331 346 361 376 391 406 421 436 451 466 481 496 511 526 541 556 571 586  1 17 33 49 65 81 97 113 129 145 161 177 193 209 225 241 257 273 289 305 321 337 353 369 385 401 417 433 449 465 481 497 513 529 545 561 577 593 609 625  1 18 35 52 69 86 103 120 137 154 171 188 205 222 239 256 273 290 307 324 341 358 375 392 409 426 443 460 477 494 511 528 545 562 579 596 613 630 647 664  1 19 37 55 73 91 109 127 145 163 181 199 217 235 253 271 289 307 325 343 361 379 397 415 433 451 469 487 505 523 541 559 577 595 613 631 649 667 685 703  1 20 39 58 77 96 115 134 153 172 191 210 229 248 267 286 305 324 343 362 381 400 419 438 457 476 495 514 533 552 571 590 609 628 647 666 685 704 723 742  1 21 41 61 81 101 121 141 161 181 201 221 241 261 281 301 321 341 361 381 401 421 441 461 481 501 521 541 561 581 601 621 641 661 681 701 721 741 761 781  1 22 43 64 85 106 127 148 169 190 211 232 253 274 295 316 337 358 379 400 421 442 463 484 505 526 547 568 589 610 631 652 673 694 715 736 757 778 799 820  1 23 45 67 89 111 133 155 177 199 221 243 265 287 309 331 353 375 397 419 441 463 485 507 529 551 573 595 617 639 661 683 705 727 749 771 793 815 837 859  1 24 47 70 93 116 139 162 185 208 231 254 277 300 323 346 369 392 415 438 461 484 507 530 553 576 599 622 645 668 691 714 737 760 783 806 829 852 875 898  1 25 49 73 97 121 145 169 193 217 241 265 289 313 337 361 385 409 433 457 481 505 529 553 577 601 625 649 673 697 721 745 769 793 817 841 865 889 913 937  1 26 51 76 101 126 151 176 201 226 251 276 301 326 351 376 401 426 451 476 501 526 551 576 601 626 651 676 701 726 751 776 801 826 851 876 901 926 951 976  1 27 53 79 105 131 157 183 209 235 261 287 313 339 365 391 417 443 469 495 521 547 573 599 625 651 677 703 729 755 781 807 833 859 885 911 937 963 989 1015  1 28 55 82 109 136 163 190 217 244 271 298 325 352 379 406 433 460 487 514 541 568 595 622 649 676 703 730 757 784 811 838 865 892 919 946 973 1000 1027 1054  1 29 57 85 113 141 169 197 225 253 281 309 337 365 393 421 449 477 505 533 561 589 617 645 673 701 729 757 785 813 841 869 897 925 953 981 1009 1037 1065 1093  1 30 59 88 117 146 175 204 233 262 291 320 349 378 407 436 465 494 523 552 581 610 639 668 697 726 755 784 813 842 871 900 929 958 987 1016 1045 1074 1103 1132  C:  0: 30 465 900 1335 1770 2205 2640 3075 3510 3945 4380 4815 5250 5685 6120 6555 6990 7425 7860 8295 8730 9165 9600 10035 10470 10905 11340 11775 12210 12645 13080 13515 13950 14385 14820 15255 15690 16125 16560 16995  1: 465 9455 18445 27435 36425 45415 54405 63395 72385 81375 90365 99355 108345 117335 126325 135315 144305 153295 162285 171275 180265 189255 198245 207235 216225 225215 234205 243195 252185 261175 270165 279155 288145 297135 306125 315115 324105 333095 342085 351075  2: 900 18445 35990 53535 71080 88625 106170 123715 141260 158805 176350 193895 211440 228985 246530 264075 281620 299165 316710 334255 351800 369345 386890 404435 421980 439525 457070 474615 492160 509705 527250 544795 562340 579885 597430 614975 632520 650065 667610 685155  3: 1335 27435 53535 79635 105735 131835 157935 184035 210135 236235 262335 288435 314535 340635 366735 392835 418935 445035 471135 497235 523335 549435 575535 601635 627735 653835 679935 706035 732135 758235 784335 810435 836535 862635 888735 914835 940935 967035 993135 1019235  4: 1770 36425 71080 105735 140390 175045 209700 244355 279010 313665 348320 382975 417630 452285 486940 521595 556250 590905 625560 660215 694870 729525 764180 798835 833490 868145 902800 937455 972110 1006765 1041420 1076075 1110730 1145385 1180040 1214695 1249350 1284005 1318660 1353315  5: 2205 45415 88625 131835 175045 218255 261465 304675 347885 391095 434305 477515 520725 563935 607145 650355 693565 736775 779985 823195 866405 909615 952825 996035 1039245 1082455 1125665 1168875 1212085 1255295 1298505 1341715 1384925 1428135 1471345 1514555 1557765 1600975 1644185 1687395  6: 2640 54405 106170 157935 209700 261465 313230 364995 416760 468525 520290 572055 623820 675585 727350 779115 830880 882645 934410 986175 1037940 1089705 1141470 1193235 1245000 1296765 1348530 1400295 1452060 1503825 1555590 1607355 1659120 1710885 1762650 1814415 1866180 1917945 1969710 2021475  7: 3075 63395 123715 184035 244355 304675 364995 425315 485635 545955 606275 666595 726915 787235 847555 907875 968195 1028515 1088835 1149155 1209475 1269795 1330115 1390435 1450755 1511075 1571395 1631715 1692035 1752355 1812675 1872995 1933315 1993635 2053955 2114275 2174595 2234915 2295235 2355555  8: 3510 72385 141260 210135 279010 347885 416760 485635 554510 623385 692260 761135 830010 898885 967760 1036635 1105510 1174385 1243260 1312135 1381010 1449885 1518760 1587635 1656510 1725385 1794260 1863135 1932010 2000885 2069760 2138635 2207510 2276385 2345260 2414135 2483010 2551885 2620760 2689635  9: 30 465 900 1335 1770 2205 2640 3075 3510 3945 4380 4815 5250 5685 6120 6555 6990 7425 7860 8295 8730 9165 9600 10035 10470 10905 11340 11775 12210 12645 13080 13515 13950 14385 14820 15255 15690 16125 16560 16995  10: 465 9455 18445 27435 36425 45415 54405 63395 72385 81375 90365 99355 108345 117335 126325 135315 144305 153295 162285 171275 180265 189255 198245 207235 216225 225215 234205 243195 252185 261175 270165 279155 288145 297135 306125 315115 324105 333095 342085 351075  11: 900 18445 35990 53535 71080 88625 106170 123715 141260 158805 176350 193895 211440 228985 246530 264075 281620 299165 316710 334255 351800 369345 386890 404435 421980 439525 457070 474615 492160 509705 527250 544795 562340 579885 597430 614975 632520 650065 667610 685155  12: 1335 27435 53535 79635 105735 131835 157935 184035 210135 236235 262335 288435 314535 340635 366735 392835 418935 445035 471135 497235 523335 549435 575535 601635 627735 653835 679935 706035 732135 758235 784335 810435 836535 862635 888735 914835 940935 967035 993135 1019235  13: 1770 36425 71080 105735 140390 175045 209700 244355 279010 313665 348320 382975 417630 452285 486940 521595 556250 590905 625560 660215 694870 729525 764180 798835 833490 868145 902800 937455 972110 1006765 1041420 1076075 1110730 1145385 1180040 1214695 1249350 1284005 1318660 1353315  14: 2205 45415 88625 131835 175045 218255 261465 304675 347885 391095 434305 477515 520725 563935 607145 650355 693565 736775 779985 823195 866405 909615 952825 996035 1039245 1082455 1125665 1168875 1212085 1255295 1298505 1341715 1384925 1428135 1471345 1514555 1557765 1600975 1644185 1687395  15: 2640 54405 106170 157935 209700 261465 313230 364995 416760 468525 520290 572055 623820 675585 727350 779115 830880 882645 934410 986175 1037940 1089705 1141470 1193235 1245000 1296765 1348530 1400295 1452060 1503825 1555590 1607355 1659120 1710885 1762650 1814415 1866180 1917945 1969710 2021475  16: 3075 63395 123715 184035 244355 304675 364995 425315 485635 545955 606275 666595 726915 787235 847555 907875 968195 1028515 1088835 1149155 1209475 1269795 1330115 1390435 1450755 1511075 1571395 1631715 1692035 1752355 1812675 1872995 1933315 1993635 2053955 2114275 2174595 2234915 2295235 2355555  17: 3510 72385 141260 210135 279010 347885 416760 485635 554510 623385 692260 761135 830010 898885 967760 1036635 1105510 1174385 1243260 1312135 1381010 1449885 1518760 1587635 1656510 1725385 1794260 1863135 1932010 2000885 2069760 2138635 2207510 2276385 2345260 2414135 2483010 2551885 2620760 2689635  18: 3945 81375 158805 236235 313665 391095 468525 545955 623385 700815 778245 855675 933105 1010535 1087965 1165395 1242825 1320255 1397685 1475115 1552545 1629975 1707405 1784835 1862265 1939695 2017125 2094555 2171985 2249415 2326845 2404275 2481705 2559135 2636565 2713995 2791425 2868855 2946285 3023715  19: 4380 90365 176350 262335 348320 434305 520290 606275 692260 778245 864230 950215 1036200 1122185 1208170 1294155 1380140 1466125 1552110 1638095 1724080 1810065 1896050 1982035 2068020 2154005 2239990 2325975 2411960 2497945 2583930 2669915 2755900 2841885 2927870 3013855 3099840 3185825 3271810 3357795 |