平行程式設計-作業1

metric

Question 1.

serial-8 cores

```
Matrix A (first 10x10):

Last element of A: 1223510749

Last element of B: 409091428

Last element of C: 1632602177

I/O time: 340.374518

CPU computation time: 34.336045

Total simulation time: 374.710564
```

parellel-1 core

```
First 10x10 of A:

1201060196 532813718 612296018 10218059 200143932 515287369 1462713077 1911306512 105510361 1537782341

1602153425 1515736683 1064554357 1707931077 1464209621 1916476823 918880151 329131259 631369417 1922707398

1642238306 758396971 676539299 360966848 1907386712 734232256 1654098462 1934542697 1971091350 46013391

1480123214 353070358 442591906 1222370246 56874573 490456633 998211855 87215591 258127432 1954854226

692635809 62835448 2103252863 337475912 1481068599 1092115252 1572741816 1498645178 512690914 1782697067

1840261459 972403451 2127369435 1122232047 1808263190 120821367 739266912 89458140 830487932 476894275

284729853 154226483 450574797 1325224059 2024483122 264498092 580405145 1318287136 17454180047 1908160011

513699019 2054179297 1704147650 1356936098 323671486 1994157792 959560361 1554961641 276561823 1388643397

1259270740 1174422727 912406986 571420177 1054139831 1637082562 1662831081 81486829 1653305863 284231440

729673076 933026248 2009365616 2135582966 1500942140 676562909 12576543 2139883014 446350889 1956491513

First 10x10 of B:

1307936020 1869917328 311672427 2030989492 953240460 1274543244 1889427262 1751140690 489982499 106081714

1692133863 2067653665 1987723594 475852061 974768622 1972680309 1233939792 2004636607 663662367 211291826

1616417966 1789335413 331685352 1849025344 1043383951 757724339 1743796356 417159299 1109807361 338885426

1258343066 763817100 622140524 1514150478 1795323846 2093904527 53026822 1155584075 2131400096 999559032

1921324023 995587256 838460951 502484284 1759964931 2001274576 349923194 1195467873 1084467741 944735666

1778489977 1868554549 74803537 803318043 927020492 1492103091 159929283 115209015 1626224631 592633699

685329983 1709085798 2036095706 171249821 274714602 1191464609 2000079565 1593085922 1765071869 997459397

1300691226 756388447 1377543664 725583239 1250694780 137385834 7644180 402212570 1598579658 1288369605

321215413 1595372127 912787420 844762720 211700532 1927669410 981370289 183676636 818728750 1020677451

237819431 1869639870 870939859 1231918497 251283912 1423873694 1988044990 929509831 2122984638 931986093

First 10x10 of C:

-1785971080 -1892236250 923968445 2041207551 1153384392 1789830613 -942826957 -632520094 595492860 1643864055

```

```
-1000680008 -711576948 -1242689345 -2111184158 -1855989053 -405810164 -2142147353 -1961199430 1295031784 2133999224

-1036311024 -1747234912 1008224651 -2084975104 -1344196633 1491956595 -897072478 -1943265300 -1214068585 384898817

-1556501016 1116887458 1064732430 -1558446572 1852198419 -1710606136 1051238677 1242799666 -1905439768 -1340554038

-1681007464 1058422704 -1353253482 839960196 -1053933766 -1201577468 1922665010 -1600854245 1597158655 -1567534563

-676215860 -1454009296 -2092794324 1925550090 -1559683614 1612924458 899196195 204667155 -1838254733 1069527974

970059836 1863312281 -1808296793 1496473880 -1995769572 1455962701 -1714482586 -1383594238 -784477380 -1389347888

1814390245 -1484399552 -1213275982 2082519337 1574366266 2131543626 967204541 1957174211 1875141481 -1617954294

1580486153 -1525172442 1825194406 1416182897 1265840363 -730215324 -1650765926 265163465 -1822932683 1304908891

967492507 -1492301178 -1414661821 -927465833 1752226052 2100436603 2000621533 -1225574451 -1725631769 -1406489690

Last element of A: 1539399490

Last element of B: 1417760769

Last element of C: -1337807037

I/O time: 340.381765

CPU computation time: 37.586832

Total simulation time: 377.968597
```

parellel-4 cores

```
First 10x10 of A:

92582347 48204032 616295958 1367044382 755306605 896431766 1686374638 1312458380 787501448 58246570

1863952333 79890945 1828076271 1179359737 1083409427 1974753712 505864578 890397252 2011991299 568062468

1792042 1506033261 445478399 1266693183 652542161 406488500 2010366206 985404490 1613496455 1821538352

1751802370 684093107 848296293 678905575 77233463 1720672784 702001127 2009919792 1639469907 1914778692

965113339 2089440379 939239675 790679280 876604344 1448531697 452505732 1809157840 873026648 1882751233

878051103 389847853 1101683870 452983874 313308569 304902126 773642915 674408803 1318474153 1360369082

120466286 677174282 1115922298 701724375 1900718671 585441252 970920123 839059834 354494593 1176910069

1082853506 784997233 1139950103 886466655 1106642905 502217452 416797640 1420658167 2014648840 632970849

1488304322 1314319018 1120824780 71399206 739417128 390097760 977534188 689733864 1001786456 558050181

2058820761 1932193202 1196191691 526776500 177780214 1407415096 631643788 1578161429 478928081 40055642

First 10x10 of B:

1349675956 92355875 16690245 2107015413 1816398324 1943861741 1412962012 1333232865 762363015 539239120

1081378736 1076903751 1200359446 981824777 1237534902 1002727058 65444933 1847405885 1671748124 598251698

1978448353 212049028 1434576591 1487469801 1944797642 1181175317 872961005 115214369 1727699010 2096518512

1157797274 401192776 1257416336 147115832 828582957 360203165 836201975 481627362 1959106289 1622596010

889706015 1962997718 1572294798 1225225057 1159908577 1462100062 208857254 765654373 1845776690 267002104

9233759 618595808 813265677 248177447 1737583168 2080433804 1410355531 1675844283 1144475453 1827442898

387247901 400853360 876934921 774611676 1064697749 2081538152 1851091159 348962445 19811556 571155169

1681837794 1133744345 337728819 1139576734 145238914 1510176524 2032707358 1622368490 781787902 1354042974

60949012 1388416282 810593229 1253575257 1324715687 1767610660 924747699 924601105 659797374 1599653752

475877667 708384654 1951702066 589817913 2106171765 1862982016 320716621 548621850 1909228845 2122625922

```

```
First 10x10 of C:

1442258303 140559907 632986203 -820907501 -1723262367 -1454673789 -1195630646 -1649276051 1549864463 597485690

-1349636227 1156794696 -1266531579 -2133782782 -1974022967 -1317486526 571309511 -1557164159 -611227873 1166314166

1980240395 1718082289 1880054990 -1540804312 -1697627493 1587663817 -1411640085 1100618859 -953771831 -376910432

-1385367652 1085285883 2105712629 826021407 905816420 2080875949 1538203102 -1803420142 -696391100 -757592594

1854819354 -242529199 -1783432823 2015904337 2036512921 -1384335537 661362986 -1720155083 -1576163958 -2145213959

887284862 1008443661 1914949547 701161321 2050891737 -1909631366 -2110968850 -1944714210 -1832017690 -1107155316

507714187 1078027642 1992857219 1476336051 -1329550876 -1627987892 -1472956014 1188022279 374306149 1748065238

-1530275996 1918741578 1477678922 2026043389 1251881819 2012393976 -1845462298 -1251940639 -1498530554 1987013823

1549253334 -1592231996 1931418009 1324974463 2064132815 -2137258876 1902281887 1614334969 1661583830 -2137263363

-1760268868 -1654389440 -1147073539 1116594413 -2011015317 -1024570184 952360409 2126783279 -1906810370 -2132285732

Last element of A: 327081768

Last element of B: 1284079707

Last element of C: 1611161475

I/O time: 86.899497

CPU computation time: 9.636476

Total simulation time: 96.225582
```

parellel-8 cores

```
First 10x10 of A:

877897265 1273857458 601262205 570247788 863036432 689489833 1213029764 751153952 1820307967 439993481

1221460973 738242836 86304116 1913113336 1847940570 1557231987 572842123 101242568 921153693 249736638

710446641 1237015098 906853520 17510845 1377477940 1646171602 1361746458 1841602801 1951621640 49422889

469502696 1522352357 1237242569 977104156 58549024 976620367 511393249 1791019086 1858416025 1230082636

1587606260 968564887 797304000 1424061262 984131933 514162271 680361310 1064215673 2011776234 906534479

1544732124 498842784 473568233 495656031 569014494 39044430 325203675 2037886294 1631887251 553680972

1567491975 272563986 790435455 1344881188 1009010287 1269892230 1205836687 1733528327 1363662512 1231341410

1219289142 1302443673 428604611 1986508623 163851469 106556929 888686418 1360026940 340856121 1493957723

1866648210 1157711724 648607002 827720839 932722813 1747334194 2087105036 1460009987 943170142 1330346039

1173454032 685618447 2123206122 1754115152 516945593 957323525 180383125 1842488049 300407776 1148799425

First 10x10 of B:

1513394211 809381774 1055248367 161790072 1968686977 563309327 1760280616 898125926 658033982 167607342

2045595527 234020175 1549964959 2131397810 631129971 1601990508 177020229 1902077051 1950380703 1775861783

85741932 238986416 324276043 1211372796 2103668973 285114029 1471553023 227782703 339132524 929368706

1042419197 1503862877 1704127866 2093687652 112963966 1823193430 900401664 708419880 832677794 2000399852

2013437841 571599310 571202914 1524778193 2067558381 583897733 436432821 444383368 1733335333 296816734

1347444793 1071423329 100186289 17345480 1488592048 844022228 267714870 1558574786 914279628 1584191291

199486905 1618888334 1623439945 2056770385 2013850892 939752010 823593752 1591894291 1662209007 379850313

1126433175 1595928598 892612392 1119600173 916688334 1299805370 714616015 973267220 525500832 708742493
```

```
96372342 1276484213 960598789 2055560883 583058175 1814475270 743549560 38289662 1294450890 718776834

2013875508 951832767 319060987 217006021 174546804 1358163936 280915151 2087056388 255943059 634016446

First 10x10 of C:

-1903675820 2083239232 1656510572 732037860 -1463243887 1252799160 -1321656916 1649279878 -1816625347 607600823

-1027910796 972263011 1636269075 -250456150 -1815896755 -1135744801 749862352 2003319619 -1423432900 2025598421

796188573 1476001514 1231129563 1228883641 -813820383 1931285631 -1461667815 2069385504 -2004213132 978791595

1511921893 -1268752062 -1353596861 -1224175488 171512990 -1495153499 1411794913 -1795528330 -1603873477 -1064484808

-693923195 1540164197 1368506914 -1346127841 -1243276982 1098060004 1116794131 1508599041 -549855729 1203351213

-1402790379 1570266113 573754522 513001511 2057606542 883066658 592918545 -698506216 -1748800417 2137872263

1766978880 1891452320 -1881091896 -893315723 -1272106117 -2085323056 2029430439 -969544678 -1269095777 1611191723

-1949244979 -1396595025 1321217003 -1188858500 1080539803 1406362299 1603302433 -1961673136 866356953 -2092267080

1963020552 -1860771359 1609205791 -1411685574 1515780988 -733157832 -1464312700 1498299649 -2057346264 2049122873

-1107637756 1637451214 -1852700187 1971121173 691492397 -1979479835 461298276 -365422859 556350835 1782815871

Last element of A: 760920588

Last element of B: 1671080781

Last element of C: -1862965927

I/O time: 44.647295

CPU computation time: 4.794189

Total simulation time: 49.234522
```

問題一（Q1）：估算程式中所需的總記憶體大小

（假設 int 為 4 bytes）

答案：

此作業中使用了三個矩陣：A、B、C，每個都是 N×NN \times N 的整數矩陣。每個元素佔 4 bytes（int 型態）。

總記憶體需求為：

3×N×N×4 bytes3 \times N \times N \times 4\ \text{bytes}

當 N=105N = 105 時：

3×105×105×4=132300 bytes≈129.3 KB3 \times 105 \times 105 \times 4 = 132300\ \text{bytes} \approx 129.3\ \text{KB}

因此，總記憶體使用量約為 **129.3 KB**。

問題二（Q2）：使用 N=105 時，請比較序列與 MPI 執行時間，哪一個比較快？請解釋原因（1 核心、4 核心與 8 核心）。

答案：

根據實驗結果：

執行版本	I/O 時間（秒）	CPU 時間（秒）	總模擬時間（秒）
序列	340.37	34.34	374.71
MPI（1 核心）	340.38	37.59	377.97
MPI（4 核心）	86.90	9.64	96.23
MPI（8 核心）	44.65	4.79	49.23

從表中可觀察到：

- **1 核心的 MPI 版本比序列版本還慢**，因為 MPI 初始化與通訊有額外開銷，在沒有實際分工的情況下反而拖慢整體效能。
- **使用 4 核心與 8 核心的 MPI 版本**，效能明顯提升。4 核心時總時間降至約 96 秒，8 核心時更降至 49 秒，約為序列時間的 **13%**，也就是效能提升了約 **87%**。
- **I/O 時間也有顯著下降**，可能與 MPI 程式中平行輸出與快取行為有關。

✅ 原因說明：

- MPI 可以將工作平均分配到多個核心，計算部分的負載大幅降低。
- 隨著核心數增加，效能也隨之提升，這證明程式具有良好的**平行伸縮性（scalability）**。
- 不過，**MPI 在低核心數時反而效能不佳**，這說明平行程式的優勢需建立在足夠的運算規模與核心數上，否則 MPI 本身的啟動與通訊開銷會造成反效果。

如果你需要我幫你整理成 Word 報告格式或圖表，也可以幫你進一步處理喔。是否也想放一張效能比較的長條圖呢？我可以幫你畫出來。

矩陣加法問題概述

我們的任務是計算兩個 NxN 矩陣 A 與 B 的和，結果存入矩陣 C，其中：

- **A、B 矩陣產生**：利用亂數產生器填入隨機整數。
- **矩陣加法運算**：針對每一個元素計算 C[i][j] = A[i][j] + B[i][j]。
- **結果輸出**：列印出 A、B、C 的前 10×10 區域，與矩陣最後一個元素。

程式版本

1. **序列版本 (matrix-s)**
   - **設計**：程式以一個迴圈依次處理每一列，動態配置每一列的記憶體以避免一次性配置整個矩陣（N=10^5 時，總元素數極大）。
   - **時間量測**：
     - **總模擬時間**：從程式開始到結束，用 `MPI_Wtime()` 計時。
     - **CPU 運算時間**：在每一列計算矩陣加法的迴圈前後使用 `MPI_Wtime()` 計算時間差，並累加後得到總 CPU 時間。
     - **I/O（通訊）時間**：定義為總模擬時間減去 CPU 運算時間。
2. **平行版本 (matrix-p)**
   - **設計**：使用 MPI 分散工作，每個 MPI 程序處理分配給自己的部分（列數根據核心數做均分，若有餘數則額外分配）。
   - **時間量測**：
     - 每個進程分別計算自己部分的 CPU 運算時間（計算矩陣加法）。
     - 每個進程也計算總模擬時間，並由主程序收集各進程的最大值作為最終的計算時間。
     - 除此之外，平行版本也額外計算了通訊時間（例如在傳送前後用 `MPI_Wtime()` 量測 MPI_Send/MPI_Recv 的開銷），以評估通訊成本。
   - **數據整合**：主程序除了彙整每一進程的部分結果（前 10×10 區塊和最後一個元素），也利用 MPI_Reduce 或點對點傳送的方式將各進程的時間數據整合起來，最後輸出全局結果。

時間量測方法

在兩個版本中，我們均利用 `MPI_Wtime()` 來進行時間量測，其做法如下：

- **總模擬時間**：程式進入主函數後立即記錄開始時間，程式結束前記錄結束時間，兩者相減即為總模擬時間。
- **CPU 運算時間**：在執行矩陣加法的內部迴圈前後分別記錄時間，將每次的運算時間累加後得到 CPU 總運算時間。
- **I/O / 通訊時間**：定義為總模擬時間減去 CPU 運算時間；而在 MPI 程式中，還特別記錄 MPI 傳送與接收操作所消耗的時間。

這樣的設計能夠幫助我們分辨純運算部分與資料傳輸/整合部分的時間，進而評估在不同核心數下，計算與通訊之間的平衡。

摘要

- Q1：總記憶體使用量為 $3 \times N^2 \times 4\ bytes$，當 $N = 10^5$ 時約為 **129.3 KB**。
- Q2：使用 MPI 時，1 核心因初始化開銷導致效能略差；但使用 4 核心與 8 核心時，效能顯著提升，**8 核心版本比序列程式快了約 87%**。這顯示 MPI 平行化在這個矩陣加總任務上具有良好伸縮性。

---

pi

Question 1.

1 core on pi-collect.out

```
Estimated Pi: 3.141

I/O time: 0.000001

CPU time: 3.378607

Total simulation time: 3.378607

Communication time: 0.000095
```

2 cores pi-collect.out

```
Estimated Pi: 3.141

I/O time: 0.000001

CPU time: 1.722814

Total simulation time: 1.722815

Communication time: 0.000311
```

4 cores on pi-collect.out

```
Estimated Pi: 3.142

I/O time: 0.000001

CPU time: 0.905071

Total simulation time: 0.905071

Communication time: 0.001816
```

8 cores on pi-collect.out

```
Estimated Pi: 3.142

I/O time: 0.000002

CPU time: 0.491151

Total simulation time: 0.491152

Communication time: 0.015167
```

1 core on pi-p_to_p.out

```
Estimated Pi: 3.142

I/O time: 0.000000

CPU time: 3.437701

Total simulation time: 3.437702

Communication time: 0.000000
```

2 cores on pi-p_to_p.out

```
Estimated Pi: 3.142

I/O time: 0.000000

CPU time: 1.770514

Total simulation time: 1.770514

Communication time: 0.000739
```

4 cores on pi-p_to_p.out

```
Estimated Pi: 3.141

I/O time: 0.000001

CPU time: 0.939963

Total simulation time: 0.939963

Communication time: 0.000594
```

8 cores on pi-p_to_p.out

```
Estimated Pi: 3.142

I/O time: 0.000001

CPU time: 0.502459

Total simulation time: 0.502460

Communication time: 0.019730
```

1 cores on pi-s.out

```
Estimated Pi: 3.142

I/O time: 0.000002

CPU time: 3.712378

Total simulation time: 3.712381
```

程式撰寫說明

我們針對蒙地卡羅估計 π 的問題，撰寫了三個版本：

1. **序列版 (pi-s.out)**
   利用標準 C 語言產生 N 個隨機點，並檢查是否落在半徑為 1 的圓內。整個計算過程均在單一程序內完成，並以 `MPI_Wtime()` 進行時間量測（雖然程式是序列版，但為了統一使用 MPI 時間函數，也引入了 MPI_Init/MPI_Finalize）。
2. **MPI 版本 (點對點: pi-p_to_p.out)**
   每個 MPI 程序根據分配到的點數，獨立產生隨機點並計算該部分計算內圓點的數量。各程序在運算前後以 `MPI_Wtime()` 量測 CPU 運算時間，並且在點對點通訊中，每個非零程序利用 `MPI_Send` 與主程序透過 `MPI_Recv` 傳送計算結果與各自的時間量測值。主程序再利用接收的數據計算出整體的估計值及各段時間（CPU、I/O 以及通訊時間）。
3. **MPI 版本 (集合通訊: pi-collect.out)**
   各程序的計算方式與點對點版相似，但在匯總數據時，主程序採用 `MPI_Reduce` 來進行集合通訊，將各程序的點數計算結果、CPU 時間、I/O 時間及總模擬時間求最大值。這種方式避免了多次傳送，降低了通訊延遲。

時間量測方法

在程式中，我們主要使用 `MPI_Wtime()` 來進行時間量測，並且分為以下幾個部分：

- **總模擬時間**：在程式一開始與結束時呼叫 `MPI_Wtime()`，相減後得到整個程式從開始到結束所花費的總時間。
- **CPU 運算時間**：在進行大量隨機點產生與計算落點是否在圓內的迴圈前後分別呼叫 `MPI_Wtime()`，以計算純運算所需時間。
- **I/O（通訊）時間**：用總模擬時間扣除 CPU 運算時間得到。
- **通訊時間（MPI 版本）**：在每次 MPI 傳送/接收資料前後分別以 `MPI_Wtime()` 量測，計算各次通訊所花費的時間，再將這些通訊時間加總或利用 MPI_Reduce 求取最大值，以反映整體通訊成本。

各個 MPI 程式中，會分別在每個程序中計算自己的運算與通訊時間，並最終透過 `MPI_Reduce` 或點對點傳送將數據整合到主程序中。
