

**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
**VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC**

—o0o—



**BÁO CÁO CUỐI KỲ**

**Một số bài tập trên Alchemi .NET Framework**

*Học phần:* Tính toán song song

Giảng viên hướng dẫn: **TS. ĐOÀN DUY TRUNG**

Sinh viên: **Nguyễn Công Hiếu - 20195016**

Lớp: **Toán tin 02 - khóa 64**

**HÀ NỘI, 01/2022**

# Lời nói đầu

Xã hội đang ngày càng phát triển nhanh chóng hơn bao giờ hết, đòi hỏi con người phải giải quyết những bài toán lớn và phức tạp trên nhiều lĩnh vực từ khoa học kỹ thuật như lượng tử, khí hậu, môi trường, vũ trụ hàng không, sinh học tế bào, cho đến kinh tế xã hội như ngân hàng, điện toán, ... Chúng là những bài toán mà từ lâu đã luôn là thách thức đối với nhân loại, song chỉ trong khoảng 1, 2 thập kỷ trở lại đây, chúng ta mới có được những kỹ thuật hay thuật toán để từng bước hoàn thiện các lời giải. Tuy nhiên, khối lượng dữ liệu cho các bài toán ngày một phình to khiến cho chỉ một chiếc máy tính là không đủ. Từ đó, các nhà khoa học đã đưa ra một giải pháp, đó là tính toán song song trên lưới với mỗi chiếc một chiếc máy tính có thể kết nối với nhau thông qua mạng và xử lý các bài toán con trên các nút của lưới. Một nhóm các nhà nghiên cứu tại Đại học Melbourne đã cùng nhau phát triển Alchemi framework (dựa trên .NET framework và chạy trên HĐH Windows) cho phép các lập trình viên có thể dễ dàng tạo ra một lưới tính toán.

Với sự chỉ bảo của thầy Đoàn Duy Trung, em xin phép được trình bày một số bài toán đơn giản được thiết kế để chạy trên lưới Alchemi. Các bài toán sau đây được thực nghiệm trên mô hình Cluster với 1 manager và 1 hoặc nhiều executor. Tuy nhiên do điều kiện không cho phép nên em chỉ có thể cài đặt 1 manager và 1 executor trên cùng 1 máy.

# Mục lục

<b>Lời nói đầu</b>	<b>i</b>
<b>1 Số chính phương</b>	<b>1</b>
1.1 Kết quả tính toán . . . . .	3
1.2 Giao diện Alchemi . . . . .	4
<b>2 Tích phân</b>	<b>5</b>
2.1 Kết quả tính toán . . . . .	7
2.2 Giao diện Alchemi . . . . .	8
<b>3 Bội chung nhỏ nhất</b>	<b>9</b>
3.1 Kết quả tính toán . . . . .	12
3.2 Giao diện Alchemi . . . . .	13
<b>4 Số Fibonacci</b>	<b>14</b>
4.1 Kết quả tính toán . . . . .	16
4.2 Giao diện Alchemi . . . . .	17
<b>Kết luận</b>	<b>18</b>
<b>Tài liệu tham khảo</b>	<b>19</b>

# Chương 1

## Số chính phương

### Đề bài

Liệt kê các số chính phương từ 1 đến  $n$ , với  $n$  nhập từ bàn phím. Phân chia đoạn  $[1, n]$  thành các đoạn để chạy trên các Executor trong lưới.

```
1 using System;
2 using System.Collections.Generic;
3 using System.Text;
4 using Alchemi.Core.Owner;
5 using Alchemi.Core;
6
7 namespace SquareNumbers
8 {
9     class Program : GApplication
10    {
11        static public GApplication App = new GApplication();
12        static public DateTime startTime;
13
14        [STAThread]
15        static public void Main(string[] args)
16        {
17            Console.WriteLine("Listing Square Numbers");
18            Console.WriteLine("-----");
19
20            int n, k;
21            Console.Write("Enter an integer: ");
22            n = Int32.Parse(Console.ReadLine());
23            Console.Write("Enter the number of works that each thread need to handle: ");
24            k = Int32.Parse(Console.ReadLine());
25
26            var clusters = (int)Math.Ceiling((double)n / k);
27            for (int i = 0; i < clusters; i++)
28            {
29                if (i == clusters - 1)
30                {
31                    App.Threads.Add(new CheckSquareNumber(i * k + 1, n));
32                }
33                else
34                {
35                    App.Threads.Add(new CheckSquareNumber(i * k + 1, i * k + k));
36                }
37            }
38        }
39    }
```

```
39 Console.WriteLine("\n<Login the Alchemi Grid>");
40 GConnection gconn = GConnection.FromConsole("localhost", "9000", "user", "user");
41 App.ApplicationName = "Square Numbers - Alchemi";
42 App.Connection = gconn;
43
44 App.Manifest.Add(new ModuleDependency(typeof(CheckSquareNumber).Module));
45 App.ThreadFinish += App_ThreadFinish;
46 App.ApplicationFinish += new GApplicationFinish(App_ApplicationFinish);
47
48 startTime = DateTime.Now;
49 Console.WriteLine("<Thread started!>");
50 App.Start();
51 Console.ReadLine();
52 }
53
54 static private void App_ThreadFinish(GThread thread)
55 {
56     CheckSquareNumber work = thread as CheckSquareNumber;
57     Console.Write("Thread {0} ({1}:{2}): ", thread.Id, work.start, work.end);
58     for (int i = 0; i < work.results.Count; i++)
59     {
60         Console.Write(work.results[i] + " ");
61     }
62     Console.WriteLine("\n");
63 }
64
65 static private void App_ApplicationFinish()
66 {
67     Console.WriteLine("Calculation finished after {0} second", DateTime.Now - startTime);
68 }
69 }
70
71 [Serializable]
72 class CheckSquareNumber : GThread
73 {
74     public int start, end;
75     public List<int> results = new List<int> ();
76
77     public CheckSquareNumber(int start, int end)
78     {
79         this.start = start;
80         this.end = end;
81     }
82
83     public override void Start()
84     {
85         var lower = (int)Math.Ceiling(Math.Sqrt(start));
86         var upper = (int)Math.Floor(Math.Sqrt(end));
87
88         for (int i = lower; i <= upper; i++)
89         {
90             results.Add(i*i);
91         }
92     }
93 }
94 }
95
```

## 1.1 Kết quả tính toán

```

C:\WINDOWS\system32\cmd.exe
Listing Square Numbers
-----
Enter an integer: 1000000000
Enter the number of works that each thread need to handle: 200000

<Login the Alchemi Grid>
Host [default=localhost] : localhost
Port [default=9000] : 9000
Username [default=user] : user
Password [default=user] : user

<Thread started!>
Thread 3 (600001:800000): 600625 602176 603729 605284 606841 608400 609961 611524 613089 614656 616225 617796 619369 620944 622521 624100 625681 627264 628849 630436 632025 633616 635209 636804 638401 640000 641601 643204 644809 646416 648025 649636 651249 652864 654481 656100 657721 659344 660969 662596 664225 665856 667489 669124 670761 672400 674041 675684 677329 678976 680625 682276 683929 685584 687241 688900 690561 692224 693889 695556 697225 698896 700569 702244 703921 705600 707281 708964 710649 712336 714025 715716 717409 719104 720801 722500 724201 725904 727609 729316 731025 732736 734449 736164 737881 739600 741321 743044 744769 746496 748225 749956 751689 753424 755161 756900 758641 760384 762129 763876 765625 767376 769129 770884 772641 774400 776161 777924 779689 781456 783225 784996 786769 788544 790321 792100 793881 795664 797449 799236

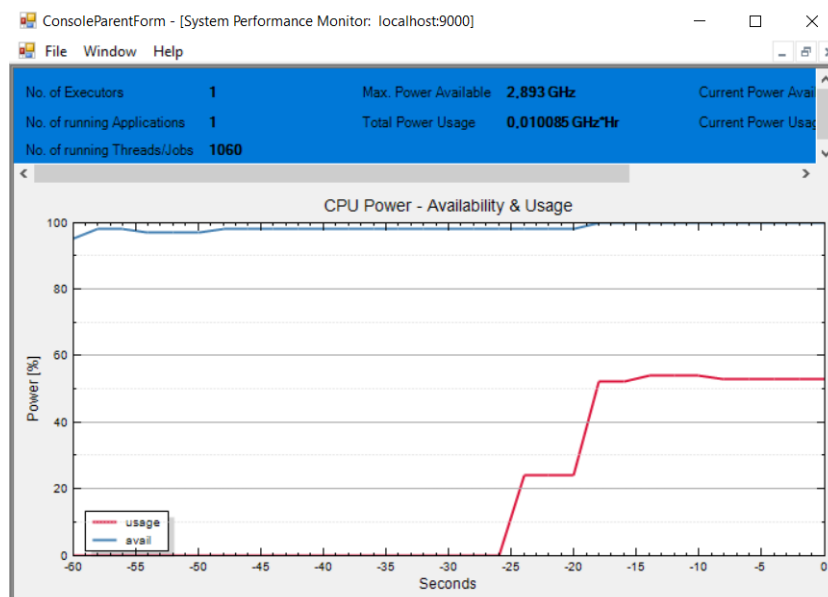
Thread 4 (800001:1000000): 801025 802816 804609 806404 808201 810000 811801 813604 815409 817216 819025 820836 822649 824464 826281 828100 829921 831744 833569 835396 837225 839056 840889 842724 844561 846400 848241 850084 851929 853776 855625 857476 859329 861184 863041 864900 866761 868624 870489 872356 874225 876096 877969 879844 881721 883600 885481 887364 889249 891136 893025 894916 896809 898704 900601 902500 904401 906304 908209 910116 912025 913936 915849 917764 919681 921600 923521 925444 927369 929296 931225 933156 935089 937024 938961 940900 942841 944784 946729 948676 950625 952576 954529 956484 958441 960400 962361 964324 966289 968256 970225 972196 974169 976144 978121 980100 982081 984064 986049 988036 990025 992016 994009 996004 998001 1000000

Thread 5 (1000001:1200000): 1002001 1004004 1006009 1008016 1010025 1012036 1014049 1016064 1018081 1020100 1022121 1024144 1026169 1028196 1030225 1032256 1034289 1036324 1038361 1040400 1042441 1044484 1046529 1048576 1050625 1052676 1054729 1056784 1058841 1060900 1062961 1065024 1067089 1069160 1071236 1073309 1075384 1077461 1079544 1081625 1083709 1085796 1087889 1089984 1092081 1094184 1096289 1098396 1100509 1102624 1104741 1106864 1108996 1111129 1113264 1115409 1117556 1119709 1121864 1124025 1126184 1128349 1130516 1132689 1134864 1137041 1139224 1141409 1143596 1145789 1147984 1150184 1152389 1154596 1156809 1159024 1161241 1163464 1165696 1167929 1170169 1172409 1174656 1176909 1179164 1181425 1183689 1185956 1188229 1190504 1192784 1195069 1197356 1199649 1201944 1204241 1206544 1208849 1211156 1213469 1215784 1218104 1220429 1222756 1225089 1227424 1229761 1232104 1234449 1236796 1239149 1241504 1243864 1246229 1248596 1250969 1253344 1255724 1258109 1260496 1262889 1265284 1267684 1270089 1272496 1274909 1277324 1279744 1282169 1284596 1287029 1289464 1291904 1294349 1296796 1299249 1301704 1304164 1306629 1309096 1311569 1314049 1316529 1319016 1321504 1323996 1326489 1328984 1331489 1333996 1336509 1339024 1341544 1344069 1346596 1349129 1351664 1354204 1356749 1359296 1361849 1364404 1366964 1369529 1372096 1374669 1377244 1379824 1382409 1384996 1387589 1390184 1392784 1395389 1397996 1400609 1403224 1405844 1408469 1411096 1413729 1416364 1418996 1421629 1424269 1426909 1429556 1432204 1434856 1437509 1440169 1442829 1445489 1448156 1450829 1453504 1456184 1458869 1461556 1464249 1466944 1469644 1472349 1475056 1477769 1480489 1483209 1485929 1488656 1491384 1494116 1496849 1499584 1502324 1505069 1507816 1510569 1513324 1516084 1518849 1521616 1524389 1527164 1529944 1532729 1535516 1538304 1541096 1543889 1546684 1549484 1552289 1555096 1557909 1560724 1563544 1566369 1569196 1572029 1574864 1577704 1580549 1583396 1586249 1589104 1591964 1594829 1597696 1600569 1603444 1606324 1609209 1612096 1614984 1617879 1620776 1623679 1626584 1629496 1632409 1635324 1638244 1641169 1644096 1647029 1649964 1652904 1655849 1658796 1661749 1664704 1667664 1670629 1673596 1676569 1679544 1682524 1685509 1688496 1691489 1694484 1697484 1700489 1703496 1706504 1709516 1712529 1715544 1718564 1721589 1724616 1727649 1730684 1733724 1736769 1739816 1742869 1745924 1748984 1752049 1755116 1758184 1761256 1764329 1767404 1770484 1773569 1776656 1779749 1782844 1785944 1789049 1792156 1795269 1798384 1801496 1804616 1807736 1810856 1813979 1817104 1820229 1823356 1826489 1829624 1832764 1835904 1839049 1842196 1845344 1848496 1851649 1854804 1857964 1861124 1864289 1867456 1870629 1873804 1876984 1880169 1883356 1886544 1889736 1892929 1896124 1899316 1902509 1905704 1908904 1912104 1915309 1918516 1921724 1924936 1928149 1931369 1934584 1937804 1941029 1944256 1947489 1950724 1953964 1957209 1960456 1963704 1966956 1970209 1973464 1976724 1980084 1983349 1986616 1989889 1993164 1996444 1999729 2003016 2006304 2009596 2012889 2016184 2019484 2022789 2026096 2029404 2032716 2036029 2039344 2042664 2045984 2049309 2052636 2055964 2059296 2062629 2065964 2069304 2072644 2075989 2079336 2082684 2086036 2089389 2092744 2096096 2099456 2102816 2106179 2109544 2112916 2116289 2119664 2123044 2126424 2129809 2133196 2136584 2139976 2143369 2146764 2150164 2153569 2156976 2160384 2163796 2167209 2170624 2174044 2177469 2180896 2184324 2187756 2191189 2194624 2198064 2201509 2204956 2208404 2211856 2215309 2218764 2222224 2225689 2229156 2232629 2236104 2239584 2243064 2246549 2250036 2253529 2257024 2260524 2264029 2267536 2271044 2274556 2278069 2281584 2285104 2288629 2292156 2295689 2299224 2302764 2306309 2309856 2313404 2316956 2320509 2324064 2327624 2331189 2334756 2338329 2341904 2345484 2349069 2352656 2356249 2359844 2363444 2367049 2370656 2374269 2377884 2381504 2385129 2388756 2392389 2396024 2399664 2403309 2406956 2410609 2414269 2417936 2421604 2425279 2428956 2432644 2436336 2440036 2443744 2447456 2451169 2454889 2458609 2462336 2466069 2469804 2473544 2477289 2481036 2484789 2488544 2492304 2496064 2500000

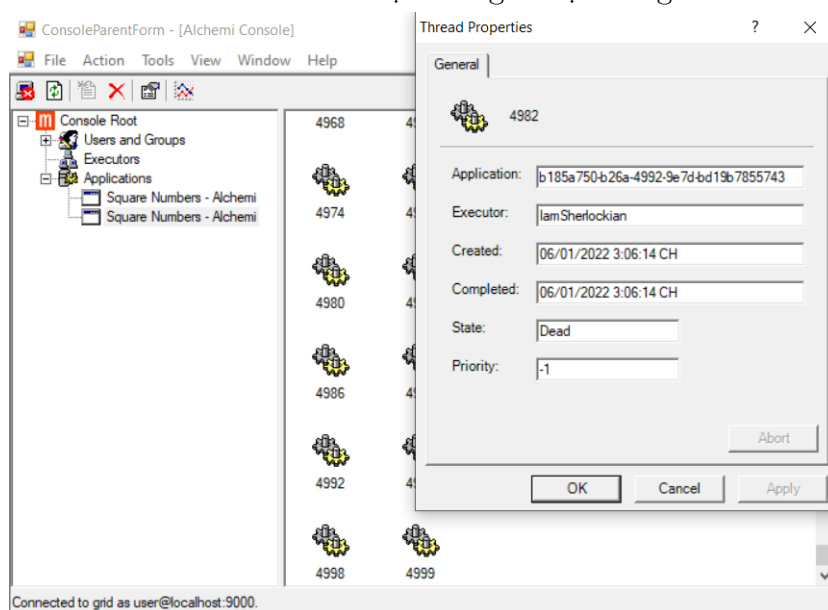
Calculation finished after 00:00:37.9150809 second

```

## 1.2 Giao diện Alchemi



Hình 1.1: Đồ thị đánh giá hiệu năng



Hình 1.2: Giao diện quản lý luồng trong Alchemi Grid

# Chương 2

## Tích phân

### Đề bài

Cho  $n$  nhập từ bàn phím. Tính gần đúng tích phân sau:

$$\int_0^n f(x)dx$$

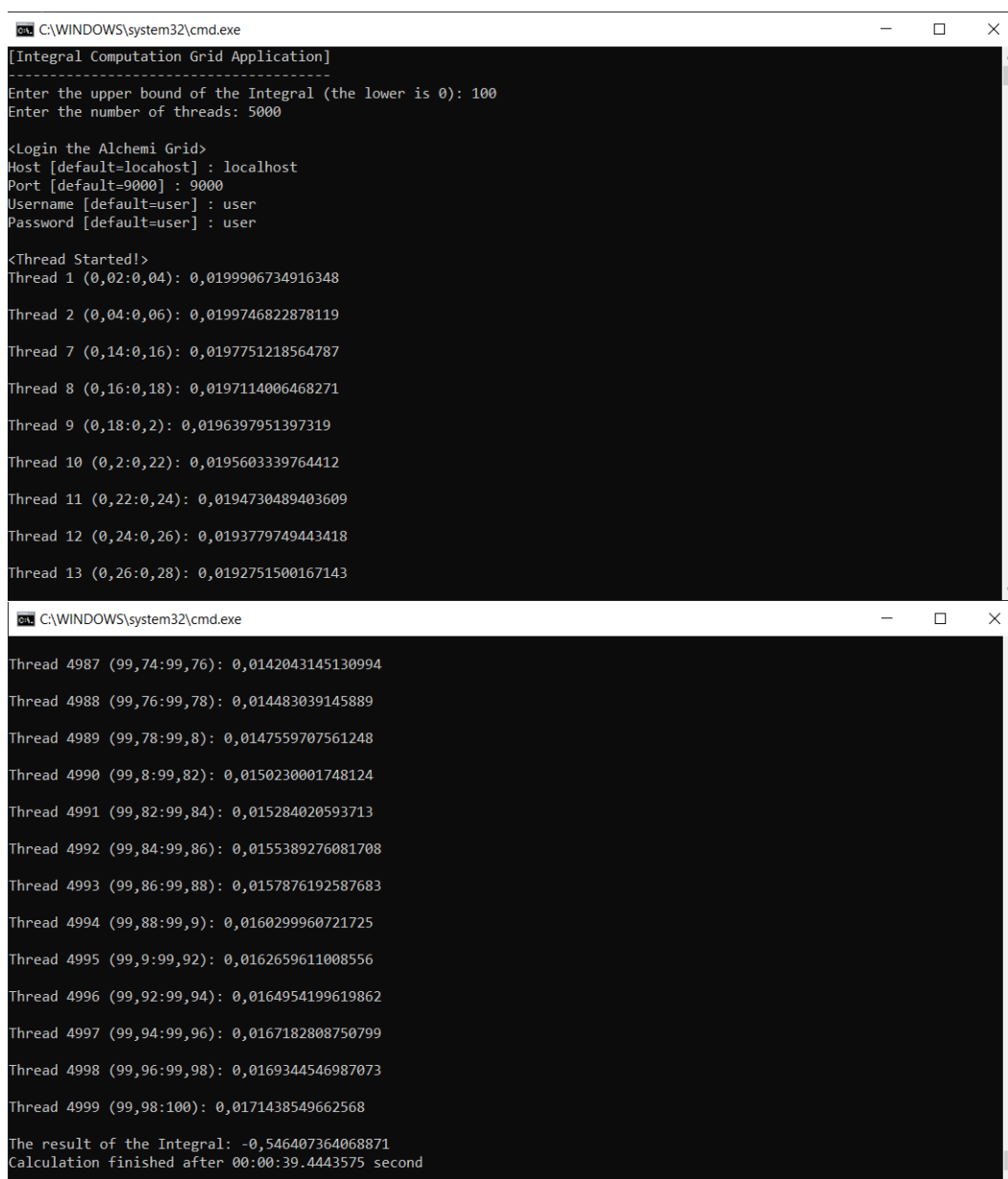
Ở đó, hàm  $f(x)$  là tùy ý. Phân chia  $[0,n]$  vào các Executor để chạy trong lưới.

```
1 using System;
2 using System.Collections.Generic;
3 using System.Text;
4 using Alchemi.Core;
5 using Alchemi.Core.Owner;
6
7 namespace IntegralCalculator
8 {
9     delegate double MathFunction(double x);
10
11     class Program
12     {
13         static public GApplication App = new GApplication();
14         static public DateTime startTime;
15         static public double Sum_res = 0;
16
17         [STAThread]
18         static public void Main(string[] args)
19         {
20             Console.WriteLine("[Integral Computation Grid Application]");
21             Console.WriteLine("-----");
22
23             Console.Write("Enter the upper bound of the Integral (the lower is 0): ");
24             var n = Double.Parse(Console.ReadLine());
25             Console.Write("Enter the number of threads: ");
26             var num_threads = Int32.Parse(Console.ReadLine());
27
28             var step = n / num_threads;
29             for (int i = 0; i < num_threads; i++)
30             {
31                 App.Threads.Add(new IntegralCal(Math.Cos, i*step, (i+1)*step));
```



```
// CHANGE function in here!
32     }
33
34     Console.WriteLine("\n<Login the Alchemi Grid>");
35     GConnection gconn = GConnection.FromConsole("localhost", "9000", "user", "user");
36     App.ApplicationName = "Integral Approximation - Alchemi";
37     App.Connection = gconn;
38
39     App.Manifest.Add(new ModuleDependency(typeof(IntegralCal).Module));
40     App.ThreadFinish += App_ThreadFinish;
41     App.ApplicationFinish += new GApplicationFinish(App_ApplicationFinish);
42
43     startTime = DateTime.Now;
44     Console.WriteLine("<Thread Started!>");
45     App.Start();
46     Console.ReadLine();
47 }
48
49 static private void App_ThreadFinish(GThread thread)
50 {
51     IntegralCal work = thread as IntegralCal;
52     Console.Write("Thread {0} ({1}:{2}): ", thread.Id, work.start, work.end);
53     Sum_res += work.result;
54     Console.Write(work.result + " ");
55     Console.WriteLine("\n");
56 }
57
58 static private void App_ApplicationFinish()
59 {
60     Console.WriteLine("The result of the Integral: {0}", Sum_res);
61     Console.WriteLine("Calculation finished after {0} second", DateTime.Now - startTime);
62 }
63 }
64
65 [Serializable]
66 class IntegralCal : GThread
67 {
68     public double start, end;
69     public double result;
70     public MathFunction Function { get; set; }
71
72     public IntegralCal(MathFunction Function, double start, double end)
73     {
74         this.Function = Function;
75         this.start = start;
76         this.end = end;
77     }
78
79     public override void Start()
80     {
81         const int m = 1000;
82         double dx = (end - start) / m;
83         double x = start;
84         double sum = 0;
85         double y = 0;
86
87         for (int i = 1; i <= m; i++)
88         {
89             y = Function(x);
90             sum += y * dx;
91             x += dx;
92         }
93         result = sum;
94     }
95 }
96 }
97
```

## 2.1 Kết quả tính toán



```
C:\WINDOWS\system32\cmd.exe
[Integral Computation Grid Application]
-----
Enter the upper bound of the Integral (the lower is 0): 100
Enter the number of threads: 5000

<Login the Alchemi Grid>
Host [default=localhost] : localhost
Port [default=9000] : 9000
Username [default=user] : user
Password [default=user] : user

<Thread Started!>
Thread 1 (0,02:0,04): 0,0199906734916348

Thread 2 (0,04:0,06): 0,0199746822878119

Thread 7 (0,14:0,16): 0,0197751218564787

Thread 8 (0,16:0,18): 0,0197114006468271

Thread 9 (0,18:0,2): 0,0196397951397319

Thread 10 (0,2:0,22): 0,0195603339764412

Thread 11 (0,22:0,24): 0,0194730489403609

Thread 12 (0,24:0,26): 0,0193779749443418

Thread 13 (0,26:0,28): 0,0192751500167143

Thread 4987 (99,74:99,76): 0,0142043145130994

Thread 4988 (99,76:99,78): 0,014483039145889

Thread 4989 (99,78:99,8): 0,0147559707561248

Thread 4990 (99,8:99,82): 0,0150230001748124

Thread 4991 (99,82:99,84): 0,015284020593713

Thread 4992 (99,84:99,86): 0,0155389276081708

Thread 4993 (99,86:99,88): 0,0157876192587683

Thread 4994 (99,88:99,9): 0,0160299960721725

Thread 4995 (99,9:99,92): 0,0162659611008556

Thread 4996 (99,92:99,94): 0,0164954199619862

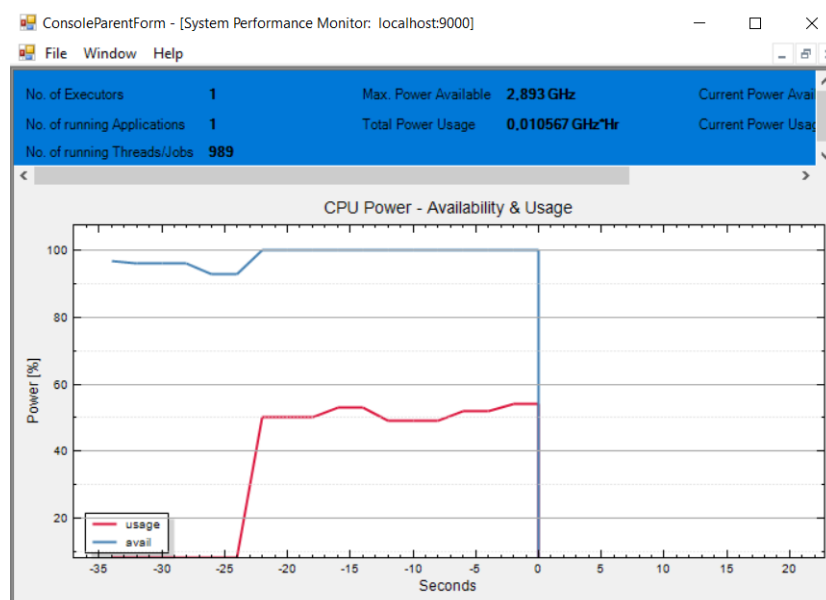
Thread 4997 (99,94:99,96): 0,0167182808750799

Thread 4998 (99,96:99,98): 0,0169344546987073

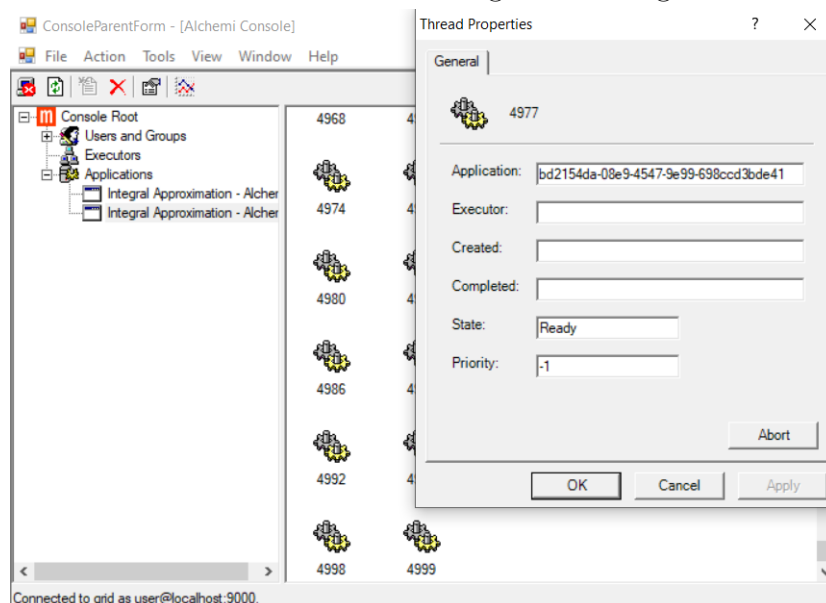
Thread 4999 (99,98:100): 0,0171438549662568

The result of the Integral: -0.546407364068871
Calculation finished after 00:00:39.4443575 second
```

## 2.2 Giao diện Alchemi



Hình 2.1: Đồ thị đánh giá hiệu năng



Hình 2.2: Giao diện quản lý luồng trong Alchemi Grid

## Chương 3

# Bội chung nhỏ nhất

### Đề bài

Nhập 1 ma trận cỡ  $m \times n$ , tìm bội số chung nhỏ nhất của từng hàng trong ma trận đó. Yêu cầu, phân chia ma trận thành nhiều ma trận nhỏ hơn với kích thước  $k \times b$ , mỗi ma trận nhỏ là 1 luồng và thực hiện trên các nút tính toán.

```
1 using System;
2 using System.IO;
3 using System.Collections.Generic;
4 using System.Text;
5 using Alchemi.Core;
6 using Alchemi.Core.Owner;
7 using Swensen;
8
9 namespace Problem_02
10 {
11     class Program : GApplication
12     {
13         static public GApplication App = new GApplication();
14         static public List<List<BigInt>> Matrix = new List<List<BigInt>>();
15         static public DateTime startTime;
16
17         static public void PrintMatrix()
18         {
19             using (TextWriter tw = new StreamWriter(@"..\..\matrix.txt"))
20             {
21                 for (int i = 0; i < Matrix.Count; i++)
22                 {
23                     for (int j = 0; j < Matrix[0].Count; j++)
24                     {
25                         tw.Write(Matrix[i][j] + " ");
26                     }
27                     tw.WriteLine();
28                 }
29             }
30         }
31
32         static public List<List<BigInt>> SubMatrix(int start_idx, int end_idx)
33         {
34             List<List<BigInt>> sendMatrix = new List<List<BigInt>>();
35             for (int i = start_idx; i <= end_idx; i++)
```

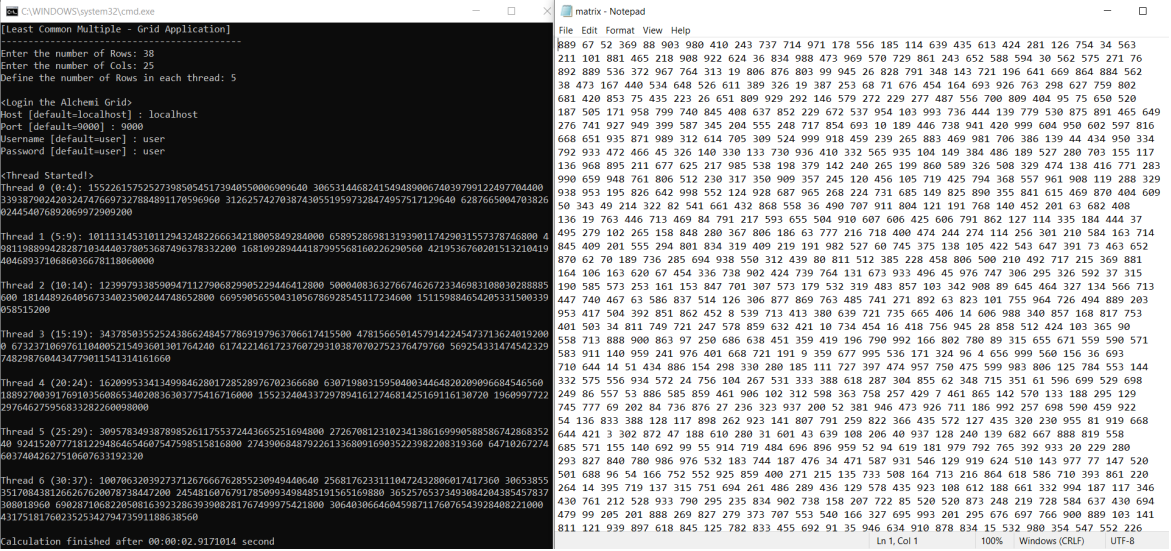
```

36     {
37         List<BigInt> temp = new List<BigInt>();
38         for (int j = 0; j < Matrix[0].Count; j++)
39         {
40             temp.Add(Matrix[i][j]);
41         }
42         sendMatrix.Add(temp);
43     }
44     return sendMatrix;
45 }
46
47 [STAThread]
48 static public void Main(string[] args)
49 {
50     Console.WriteLine("[Least Common Multiple - Grid Application]");
51     Console.WriteLine("-----");
52
53     Random rnd = new Random();
54     Console.Write("Enter the number of Rows: ");
55     var m = Int32.Parse(Console.ReadLine());
56     Console.Write("Enter the number of Cols: ");
57     var n = Int32.Parse(Console.ReadLine());
58     Console.Write("Define the number of Rows in each thread: ");
59     var k = Int32.Parse(Console.ReadLine());
60
61     for (int i = 0; i < m; i++)
62     {
63         List<BigInt> temp = new List<BigInt>();
64         for (int j = 0; j < n; j++)
65         {
66             temp.Add((BigInt)rnd.Next(1, 100)); // random numbers
67         }
68         Matrix.Add(temp);
69     }
70     PrintMatrix();
71
72     var clusters = m / k;
73     for (int i = 0; i < clusters; i++)
74     {
75         if (i == clusters - 1)
76         {
77             var sendMatrix = SubMatrix(i * k, m - 1);
78             App.Threads.Add(new LCMCalculator(sendMatrix, i * k, m - 1));
79         }
80         else
81         {
82             var sendMatrix = SubMatrix(i * k, i * k + k - 1);
83             App.Threads.Add(new LCMCalculator(sendMatrix, i * k, i * k + k - 1));
84         }
85     }
86
87     Console.WriteLine("\n<Login the Alchemi Grid>");
88     GConnection gconn = GConnection.FromConsole("localhost", "9000", "user", "user");
89     App.ApplicationName = "Least Common Multiple - Alchemi";
90     App.Connection = gconn;
91
92     App.Manifest.Add(new ModuleDependency(typeof(LCMCalculator).Module));
93     App.ThreadFinish += App_ThreadFinish;
94     App.ApplicationFinish += new GApplicationFinish(App_ApplicationFinish);
95
96     startTime = DateTime.Now;
97     Console.WriteLine("<Thread Started!>");
98     App.Start();
99     Console.ReadLine();
100 }
101
102 static private void App_ThreadFinish(GThread thread)
103 {
104     LCMCalculator work = thread as LCMCalculator;
105     Console.WriteLine("Thread {0} ({1}:{2}): ", thread.Id, work.start, work.end);
106     for (int i = 0; i < work.results.Count; i++)
107     {
108         Console.WriteLine(work.results[i] + " ");
109     }

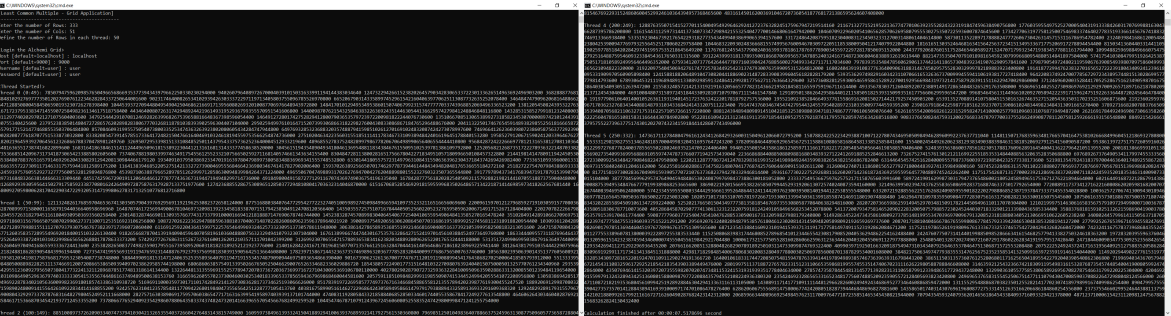
```

```
110         Console.WriteLine("\n");
111     }
112
113     static private void App_ApplicationFinish()
114     {
115         Console.WriteLine("Calculation finished after {0} second", DateTime.Now - startTime);
116     }
117 }
118
119 [Serializable]
120 class LCMCalculator : GThread
121 {
122     public int start, end;
123     public List<List<BigInt>> Matrix;
124     public List<BigInt> results = new List<BigInt>();
125
126     public LCMCalculator(List<List<BigInt>> Matrix, int start, int end)
127     {
128         this.Matrix = Matrix;
129         this.start = start;
130         this.end = end;
131     }
132
133     static public BigInt LCM(BigInt a, BigInt b)
134     {
135         BigInt res = a * b;
136         while (a != 0 && b != 0)
137         {
138             if (a > b)
139                 a %= b;
140             else
141                 b %= a;
142         }
143         return a > 0 ? res / a : res / b;
144     }
145
146     public override void Start()
147     {
148         for (int i = 0; i < Matrix.Count; i++)
149         {
150             BigInt lcm = 1;
151             for (int j = 0; j < Matrix[i].Count; j++)
152             {
153                 lcm = LCM(lcm, Matrix[i][j]);
154             }
155             results.Add(lcm);
156         }
157     }
158 }
159 }
```

### 3.1 Kết quả tính toán

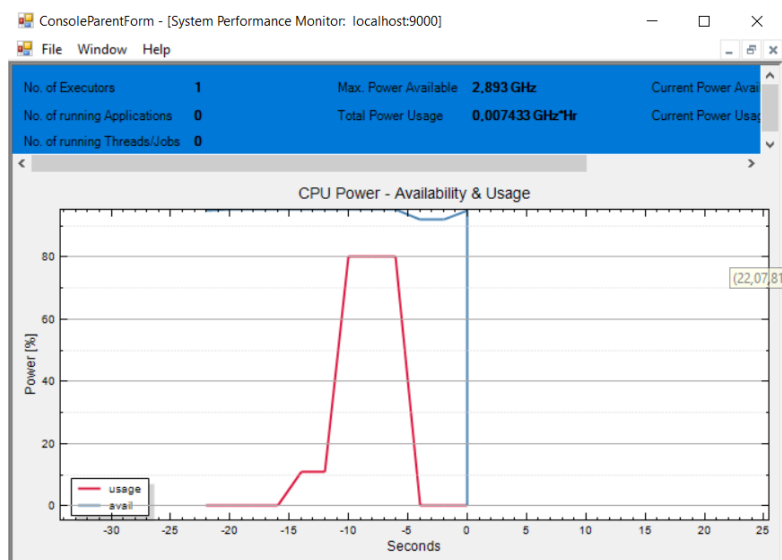


Hình 3.1: Thử nghiệm với ma trận nhỏ

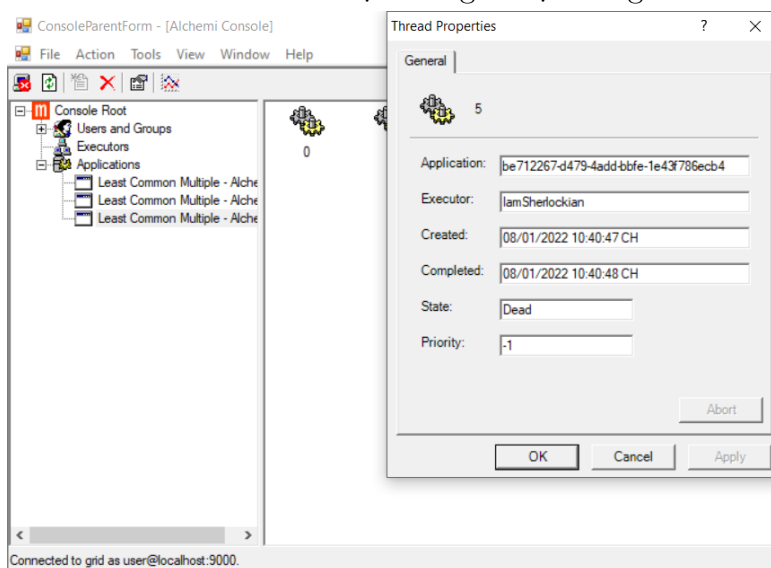


Hình 3.2: Thử nghiệm với ma trận lớn

## 3.2 Giao diện Alchemi



Hình 3.3: Đồ thị đánh giá hiệu năng



Hình 3.4: Giao diện quản lý luồng trong Alchemi Grid



# Chương 4

## Số Fibonacci

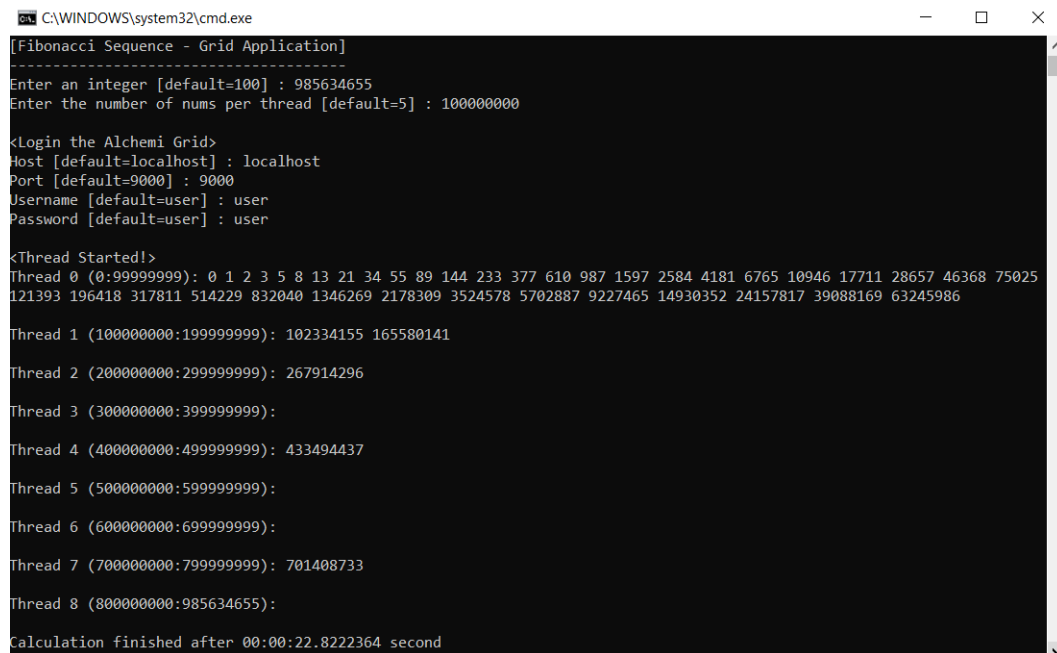
### Đề bài

Liệt kê dãy Fibonacci đến số n cho trước (n - nhập từ bàn phím).

```
1 using System;
2 using System.Collections.Generic;
3 using Alchemi.Core;
4 using Alchemi.Core.Utility;
5 using Alchemi.Core.Owner;
6 using Swensen;
7
8 namespace FibonacciSequence
9 {
10     class Program : GApplication
11     {
12         static public GApplication App = new GApplication();
13         static public DateTime startTime;
14
15         [STAThread]
16         static public void Main(string[] args)
17         {
18             Console.WriteLine("[Fibonacci Sequence - Grid Application]");
19             Console.WriteLine("-----");
20
21             var n = BigInt.Parse(Utils.ValueFromConsole("Enter an integer", "100"));
22             var k = BigInt.Parse(Utils.ValueFromConsole("Enter the number of nums per thread", "5"));
23
24             var num_threads = n / k;
25             for (BigInt i = 0; i < num_threads; i++)
26             {
27                 if (i == num_threads - 1)
28                 {
29                     App.Threads.Add(new FibListing((BigInt)(i * k), (BigInt)n));
30                 }
31                 else
32                 {
33                     App.Threads.Add(new FibListing((BigInt)(i * k), (BigInt)(i * k + k - 1)));
34                 }
35             }
36
37             Console.WriteLine("\n<Login the Alchemi Grid>");
38             GConnection gconn = GConnection.FromConsole("localhost", "9000", "user", "user");
39             App.ApplicationName = "Listing Fibonacci Sequence - Alchemi";
40             App.Connection = gconn;
41         }
42     }
43 }
```

```
42     App.Manifest.Add(new ModuleDependency(typeof(FibListing).Module));
43     App.ThreadFinish += App_ThreadFinish;
44     App.ApplicationFinish += new GApplicationFinish(App_ApplicationFinish);
45
46     startTime = DateTime.Now;
47     Console.WriteLine("<Thread Started!>");
48     App.Start();
49     Console.ReadLine();
50 }
51
52 static private void App_ThreadFinish(GThread thread)
53 {
54     FibListing work = thread as FibListing;
55     Console.Write("Thread {0} ({1}:{2}): ", thread.Id, work.start, work.end);
56     for (var i = 0; i < work.results.Count; i++)
57     {
58         Console.Write(work.results[i] + " ");
59     }
60     Console.WriteLine("\n");
61 }
62
63 static private void App_ApplicationFinish()
64 {
65     Console.WriteLine("Calculation finished after {0} second", DateTime.Now - startTime);
66 }
67
68
69 [Serializable]
70 class FibListing : GThread
71 {
72     public BigInt start, end;
73     public List<BigInt> results = new List<BigInt>();
74
75     public FibListing(BigInt start, BigInt end)
76     {
77         this.start = start;
78         this.end = end;
79     }
80
81     public bool IsPerfectSquare(BigInt x)
82     {
83         BigInt squ = BigInt.Sqrt(x);
84         return (squ * squ == x);
85     }
86
87     public bool IsFibonacci(BigInt x)
88     {
89         return IsPerfectSquare(5 * x * x + 4) || IsPerfectSquare(5 * x * x - 4);
90     }
91
92     public override void Start()
93     {
94         for (var i = start; i <= end; i++)
95         {
96             if (IsFibonacci(i) == true)
97             {
98                 results.Add(i);
99             }
100         }
101     }
102 }
103 }
104
```

## 4.1 Kết quả tính toán



```
C:\WINDOWS\system32\cmd.exe
[Fibonacci Sequence - Grid Application]
-----
Enter an integer [default=100] : 985634655
Enter the number of nums per thread [default=5] : 100000000

<Login the Alchemi Grid>
Host [default=localhost] : localhost
Port [default=9000] : 9000
Username [default=user] : user
Password [default=user] : user

<Thread Started!>
Thread 0 (0:99999999): 0 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025
121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986

Thread 1 (100000000:199999999): 102334155 165580141

Thread 2 (200000000:299999999): 267914296

Thread 3 (300000000:399999999):

Thread 4 (400000000:499999999): 433494437

Thread 5 (500000000:599999999):

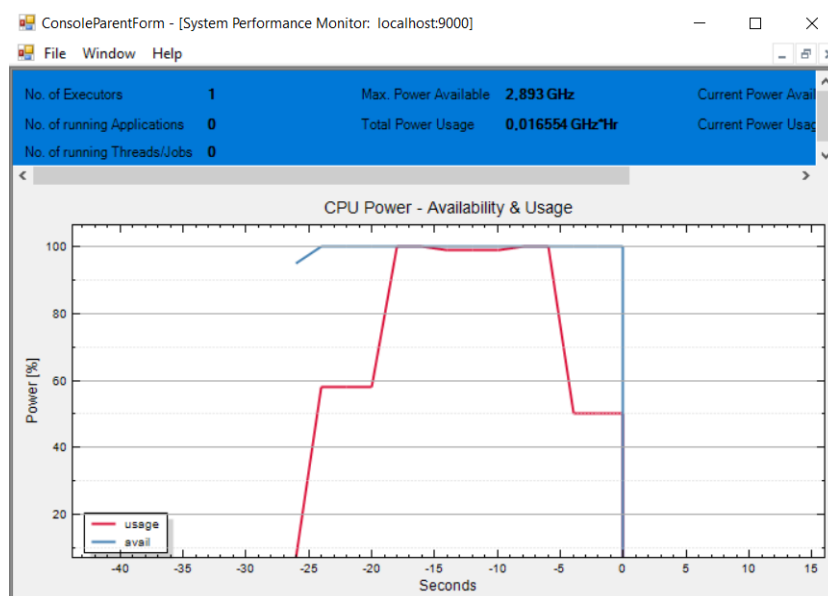
Thread 6 (600000000:699999999):

Thread 7 (700000000:799999999): 701408733

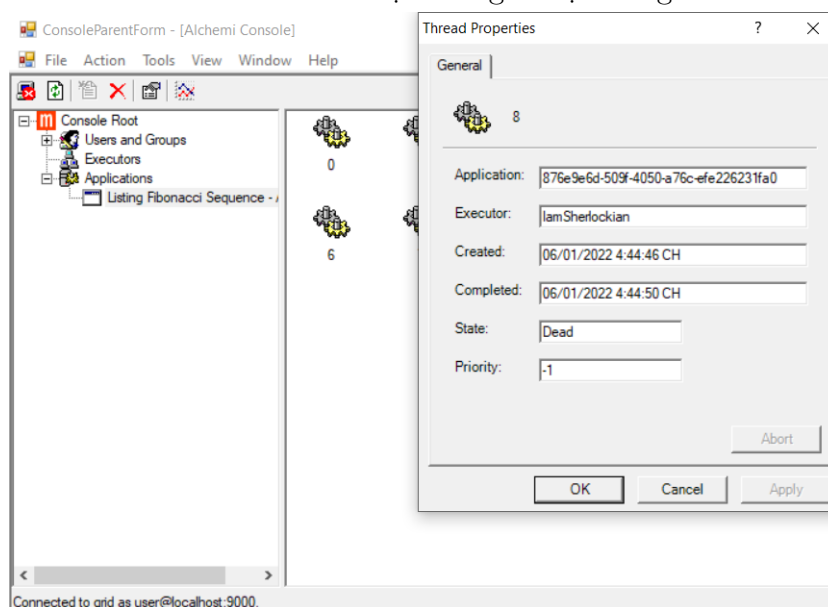
Thread 8 (800000000:985634655):

Calculation finished after 00:00:22.8222364 second
```

## 4.2 Giao diện Alchemi



Hình 4.1: Đồ thị đánh giá hiệu năng



Hình 4.2: Giao diện quản lý luồng trong Alchemi Grid

# Kết luận

Với sự hướng dẫn tận tình của thầy Đoàn Duy Trung, em đã hoàn thành bài báo cáo này với hi vọng bản thân sẽ được tiếp cận tới một trong những kỹ thuật quan trọng nhất trong lĩnh vực toán-tin, công nghệ thông tin, tối ưu, ... Và giúp rèn luyện tư duy song song hóa, không chỉ có thể đưa ra lời giải cho một bài toán mà còn có thể phân tích, chia thành nhiều bài toán nhỏ để giải quyết đồng thời.

Do thời gian có hạn và khả năng còn hạn chế, bài báo cáo có thể không được hoàn chỉnh và sai sót là không tránh khỏi. Rất mong thầy và các bạn đọc bài báo cáo này có thể đóng góp ý kiến. Em xin cảm ơn!

# Tài liệu tham khảo

- [1] Slide lập trình Alchemi của thầy Đoàn Duy trung.
- [2] <https://sourceforge.net/projects/alchemi/>
- [3] <https://arxiv.org/ftp/cs/papers/0402/0402017.pdf>
- [4] <https://www.codeproject.com/Articles/36323/BigInt>