**تمرین درس سیستم های توزیع شده – مبحث MPI**

نیمسال تحصیلی 951

**مرتضی ذاکری - 95723088**

**تعداد مسئله ها: 2**

# تعیین مقدار π با استفاده از روش مونت کارلو

با استفاده از روش مونت کارلو که یک روش انتگرال عددی می باشد مقدار عدد PI را تعیین نمایید. N نقطه تصادفی در مربع با مشخصات x بین 1- تا 1 و y بین 1- تا 1 در نظر بگیرید و مقدار پی را از طریق رابطه زیر محاسبه نمایید.

**π = 4 n/N**

 nتعداد نقاطی است که درون دایره ای به شعاع یک قرار می گیرند.

* 1. در ابتدا یک کد سریال بنویسید که مقدار پی را محاسبه نماید.
	2. سپس با استفاده از MPI کدی موازی بنویسید و مطمئن شوید که کدی که نوشته اید همان نتیجه کد سریال را باز میگرداند.
	3. زمان را برای N=108 محاسبه نمایید هنگامی که از 1 و 2 و 4 و 16 پردازنده استفاده می نمایید. (برای این کار میتوانید از MPI\_WTIME استفاده نمایید و یا از دستور موجود در UNIX به صورت time ./mc\_pi استفاده نمایید.). در گزارش خود مطرح نمایید که آیا زمان initialization را در نظر گرفته اید یا خیر؟

## روش مونت کارلو

در ابتدا روش مونت کارلو برای محاسبه عدد PI را مطرح می کنیم. در این روش دایره ای را در نظر گرفته که داخل یک مربع محاط شده است. اگر طول ضلع مربع را 2R در نظر بگیریم آن گاه شعاع دایره برابر R خواهد بود و نسبت مساحت دایره به مساحت مربع برابر خواهد بود با:

πR2 / (2R)2 = π/ 4

تعدادی نقطه تصادفی تولید می کنیم به طوری که در محدوده ی داخلی مربع قرار گیرند. حال اگر نسبت محاسبه شده در رابطه فوق را برابر با نسبت تعداد نقاط تصادفی که داخل دایره قرار گرفته اند (n) به کل نقاط تولید شده (N) قرار دهیم، می توانیم مقدار π را از رابطه زیر با تقریب خوبی محاسبه کنیم. بدیهی هر چه تعداد نقاط تصادفی تولید شده بیش تر باشد آن گاه دقت محاسبه عدد π نیز بیش تر خواهد بود.

π/ 4 = n / N => π= 4n / N

## راه حل سریال

کد شکل ‏1‑1 مقدار π را به صورت سریال حساب می کند و باز می گرداند. چند خروجی نمونه حاصل از اجرای کد با مقادیر مختلف برای تعداد نقاط نیز در زیر آمده است.







شکل ‏1‑1محاسبه PI به صورت سریال توسط روش مونت کارلو در زبان C

علاوه بر این می توان چندین بار تابع فوق را در یک حلقه فراخوانی کرد و از مقادیر تولید شده میانگین گرفت تا دقت افزایش یابد. در زیر این کد آمده است:



## راه حل موازی با استفاده از MPI

برای حل به صورت موازی می توان از این ایده استفاده کرد که هر یک از Slave ها مقدار PI را جداگانه محاسبه کنند. Master نیز خود یک بار این مقدار را محاسبه کند و سپس Slave ها مقدار محاسبه شده را به Master ارسال نمایند، Master این مقادیر را دریافت کرده، میانگین آن ها را محاسبه و نتیجه را در خروجی چاپ کند. کد راه حل ذکر شده در فایل parallel\_pi\_calc.c آمده است. برای اجرای برنامه ابتدا آن را کامپایل و سپس اجرا می کنیم:

* **$ sudo mpicc** parallel\_pi\_calc.c -o parallel\_pi\_calc
* **$ mpirun -f hosts -n 2** ./parallel\_pi\_calc

خروجی نهایی برنامه نیز برای N=10^8 و 100 دور محاسبه، به صورت زیر است:

* PI = 3.14225600

که با دقت دو رقم اعشار دقیقا برابر با مقدار خروجی کد سریال و نیز مقدار واقعی عدد PI است. همچنین زمان اندازه گیری شده برای N=10^8 هنگامی که از 1 و 2 پردازنده استفاده می کنیم به ترتیب برابر است با:

* **$ time ./parallel\_pi\_calc**
* real 0m1.075s
* user 0m0.744s
* sys 0m0.000s
* **$ time mpirun -f hosts -n 1 ./parallel\_pi\_calc**
* real 0m0.981s
* user 0m0.720s
* sys 0m0.008s
* **$ time mpirun -f hosts -n 2 ./parallel\_pi\_calc**
* real 0m2.134s
* user 0m1.064s
* sys 0m0.064s

بدیهی است که زمان Initialization نیز در اجراهای فوق در نظر گرفته شده است. با وجود این که سیستم بنده امکان اجرا بر روی تعداد 4 و یا بیش تر پردازنده را نداشت ولی به راحتی می توان زمان این اجرا را نیز مشابه آن چه در بالا بیان شده به دست آورد.

**\* فایل کد هر دو بخش سریال و موازی پیوست شده است.**

# پردازش آرایه با استفاده از استراتژی Master-Slave

ابتدا یک کد سریال بنویسید که این کار را انجام دهد. سعی کنید یک بار به صورت سطری و بار دیگر به صورت ستونی آرایه را پیمایش نمایید. در هر دو روش مقدار زمان لازم برای انجام اینکار را محاسبه نمایید و دلایل اختلاف را بیان نمایید.

**پیمایش سطری**

real 0m0.002s

user 0m0.000s

sys 0m0.000s

**پیمایش ستونی**

real 0m0.011s

user 0m0.000s

sys 0m0.000s

چون ماتریس ها در زبان C به صورت سطری در حافظه ذخیره می شوند. پیمایش سطری زمان کمتری نسبت به پیمایش ستونی می گیرد.

با استفاده از روش Master-Salve کد موازی بنویسید به طوری که Master آرایه را مقداردهی اولیه می نماید و سپس آرایه را بین slave ها توزیع می نماید. سپس slave ها محاسبات را انجام می دهند و داده ها در نهایت توسط master جمع آوری می شود تا خروجی نهایی پدید آید. در این قسمت به سوالات زیر نیز پاسخ دهید.

1. آیا بهتر نیست Master قسمتی از آرایه را نگه دارد تا خود قسمتی از پردازش را انجام دهد؟

با توجه به این که Master در حین محاسبات Slave ها، کاری برای انجام دادن ندارد در صورت کم بودن تعداد Salveها بهتر است این کار انجام شود.

1. شما چگونه آرایه را بین پردازنده ها تقسیم می کنید؟(روش خود را به طور مختصر شرح دهید)

با استفاده از تابع MPI\_Scatter و از روی تعداد Slaveها یعنی world\_size -1 هر سطر آرایه را بین Slaveها تقسیم می کنیم. سپس با استفاده از تابع MPI\_Gather نتایج محاسبه شده را توسط Master جمع آوری می نماییم. این کار را برای هر سطر ماتریس انجام می دهیم و نهایتا در پردازه Master نتیجه را چاپ می کنیم.

1. شما چگونه می توانید کاری کنید که تمامی پردازنده ها به یک میزان بار اجرایی داشته باشند؟ (Load Balance Between Processors)

با توجه به روش موازی سازی که در قسمت 2 مطرح شد، می توان ادعا نمود بار کاری تمامی پردازنده ها (لااقل تمامی Slaveها به یک میزان است) زیرا ارایه را به صورت مساوی بین همه Slaveها تقسیم نموده ایم.

**\* فایل کد هر دو بخش سریال و موازی پیوست شده است.**

**\*\*\***