

به نام خدا

مقدمه :

در برنامه نوشته شده تابع job در پوشه worker همان process های ما هستند که در آن تابع heavy\_computational\_function اجرا میشود. فرض کردیم request ها ما یعنی همان ورودی های تابع heavy\_computational\_function متن هایی از ۱ تا ۱۰۰۰ است که حدودا در ۳۰ ثانیه ۱۰۰۰ request به تابع ما داده میشود. ( در یک حلقه while) هر پروسس برای خود یک دیکشنری cache دارد. Share memory ما یک لیست است که index این لیست بیانگر همان ورودی تابع heavy\_computational\_function است یعنی این لیست ۱۰۰۰ عضو دارد. و عضو ها به فرمت time:value هستند که time زمان قرار گرفتن این record در مموری است و value جواب تابع heavy\_computational\_function برای index این خانه است.

مراحل اجرا برنامه :

## 1 - مرحله start و شروع به کار :

در این مرحله ما یک عدد تصادفی بین ۱ تا ۹۰۰ پیدا میکنیم و تا ۱۰۰ عدد بعدی آن را به عنوان ورودی هایی در نظر میگیریم که قرار است در خود procecc کش شوند. **نکته :** طبیعی است که این سناریو اشتباه است و باید در مرحله اجرا این مقادیر به صورت اتوماتیک پیدا شوند یعنی پر استفاده ترین ورودی ها برای تابع heavy\_computational\_function در این process .

## 2 - مرحله در حال انجام کار ( در این مرحله cache ها پر شده اند و برنامه به حالت عادی در حال کار است )

- نکته برای آپدیت کردن cache : در آپدیت کردن خود کش پروسس میتواند مقدار به دست آورده را توسط یک ترد دیگر به share memory داد تا آن هم به روز شود .
- از نظر من هیچ اختلالی برای خواندن و نوشتن هم زمان پروسس ها روی shared memory ایجاد نمیشود ولی میتوان کاری کرد اگر دو پروسس به صورت هم زمان یک مقداری که داخل shared memory منقضی شده است را بخواهند هر دوی آن ها این مقدار را مجدد حساب نکنند بلکه یکی حساب بکند و مموری را آپدیت کند و دیگری آن را از مموری بخواند.

- نکته برای cache : ممکن است در طول زمان ثابت شود که ورودی برای تابع heavy\_computational\_function وجود دارد که پر استفاده تر از دیگران است و در cache نیست باید این رو در نظر بگیریم و ان را تغییر دهیم.

### 3- مرحله متوقف کردن :

در این حالت بهتر است مقادیر پر استفاده این پروسس توسط تابع heavy\_computational\_function ضخیره شود تا برای روشن کردن دوباره این مقادیر را داشته باشیم .  
هر چند که این مقادیر ممکن است تغییر کنند و آن هایی که پر استفاده تر هستند باز جایگزین شوند.

و در آخر  
**شاید** بهتر بود ۱۰ پروسس هر کدام ۱۰۰ مقدار را cache میکردند و در صورت نیاز به یک مقدار که در cache آن ها نبود به پروسس های دیگر request می داد و از آن ها مقدار را میگرفت.