۱- در برنامه‌نویسی هم‌روند با استفاده از منابع یکسان سعی می‌شود با بهره‌گیری از waiting timeهای برنامه‌های مختلف (مثل زمان‌های طولانی I/O و غیره) و switch کردن تمرکز cpu بین برنامه‌های مختلف به نحوی از cpu استفاده شود که برنامه‌ها به صورت هم‌زمان با هم و به صورت efficient جلو بروند. اما در برنامه‌نویسی موازی با استفاده از چندین منبع مختلف چندین برنامه را به صورت موازی اجرا می‌کنیم. شباهت آن‌ها جلو رفتن نسبی برنامه‌ها به صورت یکسان و تفاوت آن‌ها در میزان منابع و نحوه‌ی استفاده از آن‌ها در سیستم است.

۳-

الف) زمان اجرای برنامه‌ی سریال با write و calculation هم‌زمان: 40.136369
می‌توان خروجی برنامه را در حافظه اصلی (رم) ذخیره کرد. چون نوشتن به hard disk به صورت sequential و نه direct access انجام می‌شود، بهتر است که تمامی خروجی‌ها یک‌جا و در یک حلقه for روی hard disk انجام شود تا کم‌ترین زمان صرف نوشتن به فایل شود. این تکنیک منجر به خروجی زیر برای برنامه می‌شود.

*زمان اجرای برنامه‌ی سریال با مجزاکردن قسمت write خروجی:

Calc Time: 22.853073, Program Time: 24.330994
Program Time Speedup: 1.756 (به نسبت قسمت الف)

با مجزاکردن قسمت write خروجی و محاسبات و استفاده از parallel for سطری:

Calc Time: 12.830069, Program Time: 14.492297
Calc Time Speedup: 1.781, Program Time Speedup: 1.678 (به نسبت قسمت *)

با مجزاکردن قسمت write خروجی و محاسبات و استفاده از parallel for تودرتو:

Calc Time: 10.047349, Program Time: 11.590001
Calc Time Speedup: 2.274, Program Time Speedup: 2.1 (به نسبت قسمت *)

ب)

Calculation Time: 7.936901, Program Time: 9.560519
Calc Time Speedup: 1.265, Program Time Speedup: 1.212

به نظر می‌رسد که openmp با استفاده از task می‌تواند قسمت‌های موازی برنامه را به خوبی صف‌بندی کند تا در زمان‌های optimal اجرا شوند. به همین دلیل حتی زمان بهتری نسبت به زمان parallel for تو در تو خواهیم داشت.
در صورتی که سیستم کاربران بیش‌تری داشته‌باشد و هسته‌های کم‌تری از پردازنده قابل دسترسی باشند، بهتر است از همان parallel for استفاده شود تا overhead الگوریتم زمان‌بندی openmp موثر واقع نشود.

پ) با استفاده از حالت قبل و موازی‌سازی محاسبات، pipeline کردن بسیار پیچیده و inefficient می‌شود. چرا که نوشتن به فایل به صورت sequential انجام می‌شود و حتما باید ترتیب نوشتن به فایل‌ها حفظ شود. نتیجتا نمی‌توان آن‌ها را به عنوان قسمت‌های مختلف یک pipeline در نظر گرفت که می‌توانند بدون تداخل اجرا شوند. برای اجرایی‌ساختن الگوریتم pipeline مجبوریم تا جفت محاسبات و نوشتن به فایل را به صورت سری انجام دهیم اما هر دوی این‌ها را در تسک‌هایی موازی انجام دهیم تا با synchronize کردن آن‌ها (که چه تعداد ریزالت برای نوشتن به فایل داریم) بتوانیم یک pipeline داشته‌باشیم. که در این صورت نتیجه‌ی برنامه به صورت زیر خواهد بود:

Calc time: 22.407324, Program Time: 25.805839
Program Time Speedup: 0.94