|  |  |  |  |
| --- | --- | --- | --- |
| Solution Type | Number of Slaves | Execution Time | Explain the result |
| Sequential Solution |  | 28.721766 | חישוב רגיל וסדרתי |
| Static Task Pool | 2 | 26.578166 | חל שיפור קל עקב חילוק עבודה שכל תהליך עושה חצי מהחישובים אך עדיין יש מספר רב של חישובים לעשות |
| Static Task Pool | 4 | 20.014541 | בחילוק עבודה זה ניתן לראות שיפור משמעותי ורציני לזמן הריצה מכיוון שהכמות מתחלק באופן שווה בין ארבעה תהליכים, דבר שמקטין את כמות החישוב פי רבע ולכן תוצאה סופית של כל תהליך מתקבלת מהר יותר ובכך המאסטר יכול לחשב את החישוב הכללי מהר בהרבה. |
| Static Task Pool | 10 | 22.607313 | ניתן לראות שכאשר אנו מחלקים את התוכנית שלנו ל10 תהליכים שונים אנחנו אכן משפרים את זמן הריצה שלנו משמעותית אך חילוק זה איטי יותר מאשר הרצת התוכנית עם ארבעה תהליכים. מקרה משונה זה קורה עקב שליחה מרובה יותר של הודעות בין התהליכים למאסטר. כל שליחת הודעה לוקחת זמן ומעריכה את זמן הריצה. |
| Dynamic Task Pool | 2 | 28.472553 | ניתן לראות שאין הרבה הבדל בין הדינאמי לסדרתי ואף איטי יותר מהסטטי. מקרה זה קורה בגלל האופן בו אנחנו מתקשרים עם התהליכים דרך המאסטר. בדינאמי יש הרבה הודעות שנשלחות כדי לתקשר ולהבין אם העבדים צריכים עוד לעבוד והאם המאסטר צריך עוד לשלוח מספרים. כל ההודעות האלה הם עיקוב רציני בזמן ריצה בעיקר כאשר אין לנו הרבה תהליכים ולכן היתרון של הדינאמי לא מורגש כאשר יש לנו 2 תהליכים. החיסרון של עיקוב שליחת ההודעות גובר על היתרון בשיטה |
| Dynamic Task Pool | 4 | 11.347223 | בדוגמא זו אנו רואים את היתרון של התכנות המקבילי בשיטה הדינאמית בצורה מובהקת. כאשר יש לנו מספר אידאלי של תהליכים לסוג חישוב שאנו צריכים לבצע החישובים מתבצעים מאוד מהר וגם חילוק העבודה של המאסטר לעבדים מתבצע בקצב גבוה. זה לא מספר רב מדי של תהליכים שיכול לגרום למאסטר לעיקוב בגלל יותר מדי תהליכים לנהל. |
| Dynamic Task Pool | 10 | 25.2266124 | ניתן לראות שכאשר אנחנו נותנים למאסטר לנהל 10 תהליכים שונים הזמן ריצה מתארך והיעילות של המאסטר בניהול התהליכים קטן. דבר זה תלוי בעיקר בכוח העיבוד של המחשב שמריץ את התוכנית הנ"ל. ככל שהמחשב יהיה "חזק" יותר כך המאסטר יוכל לנהל יותר תהליכים במקביל והזמן יקטן |