GUY LESHEM

קורס מיחשוב מקבילי ומבוזר

הרצאה 1: מבוא

אדמיניסטרציה

- הרצאות פעם בשבוע בימי ראשון.
- שעה 17:00 עד 20:00

דרישות

- שפת C (program in C).
- מבנה נתונים (Data Structures).
- ארכיטקטורת מחשבים (Computer Architecture).

מדיניות מתן ציונים

- 50% בחינה סופית
- 50% עבודות בית (על 4 מערכות שונות: MPI, openMP, Matlab, IntelMKL)

פרטים ליצירת קשר

- ד"ר גיא לשם
- Email: leshemg@cs.bgu.ac.il

חומרי קורס

On-line Books:

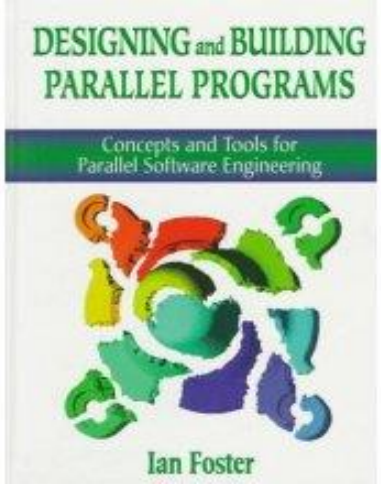
- *Ian Foster, Designing and Building Parallel Programs*

  http://www-unix.mcs.anl.gov/dbpp/

Geoffrey Fox, Parallel Computing Works.

www.npac.syr.edu/copywrite/pcw

- For this week, start reading Chapter 1



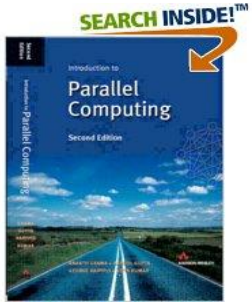
חומרי קורס

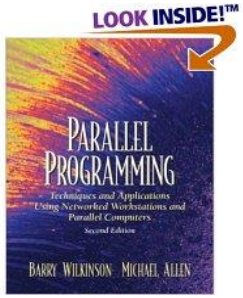
On-line Books:

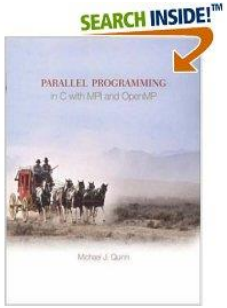
- an introduction to parallel algorithms
- http://www.cs.utah.edu/~hari/teaching/bigdata/book92-JaJa-parallel.algorithms.intro.pdf

Offline Books:

- *Vipin Kumar et al., Introduction to Parallel Computing, Benjamin, 2004.*
- *Wilkinson B. and Allen Michael., Parallel Programming, Prentice-Hall 1999.*
- *Quinn Michael., Parallel Programming in C with MPI and OpenMP, McGraw Hill, 2003*







סדר ההרצאות

הקורס יתחלק ל-3 נושאים:

א. מערכות מקביליות:

- נושאים בתכנות מקבילי 1.
- נושאים בתכנות מקבילי 2.
- ארכיטקטורת מקביליות.
- עיצוב אלגוריתם מקביליים.

ב. מודלים לתכנות מקבילי:

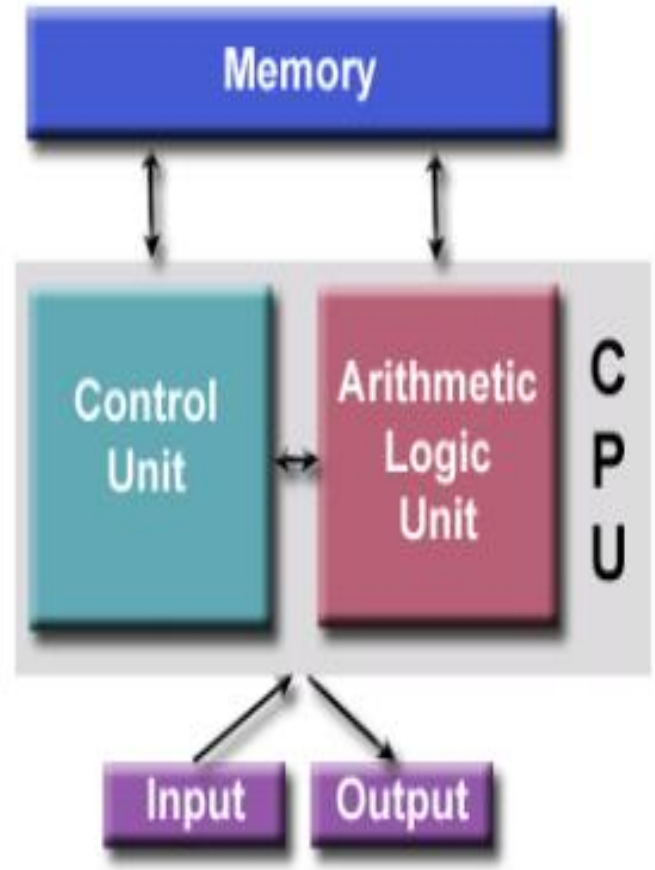
- ספריית MPI לשפת C.
- ספריית OpenMP לשפת C.
- MATLAB מקבילי.
- ספריית Intel: Math Kernel Library עבור Visual Studio לשפת C++.

ג. אלגוריתמים מקביליים:

- מיונים מקביליים - מיונים פנימיים המבוססים על השוואה של מפתחות.
- כפל מטריצה בווקטור - עיצוב, ניתוח, ויישום של שלוש תוכניות מקביליות.
- שיטת מונטה-קרלו - שיטה לפתרון בעיות חישוביות באמצעות מספרים אקראיים.
- פתרון מערכות ליניאריות - שיטות מקבילות לפתרון מערכות ליניאריות.
- הנפה של ארטוסתנס – שיטות מקביליות למציאת מספרים ראשונים.

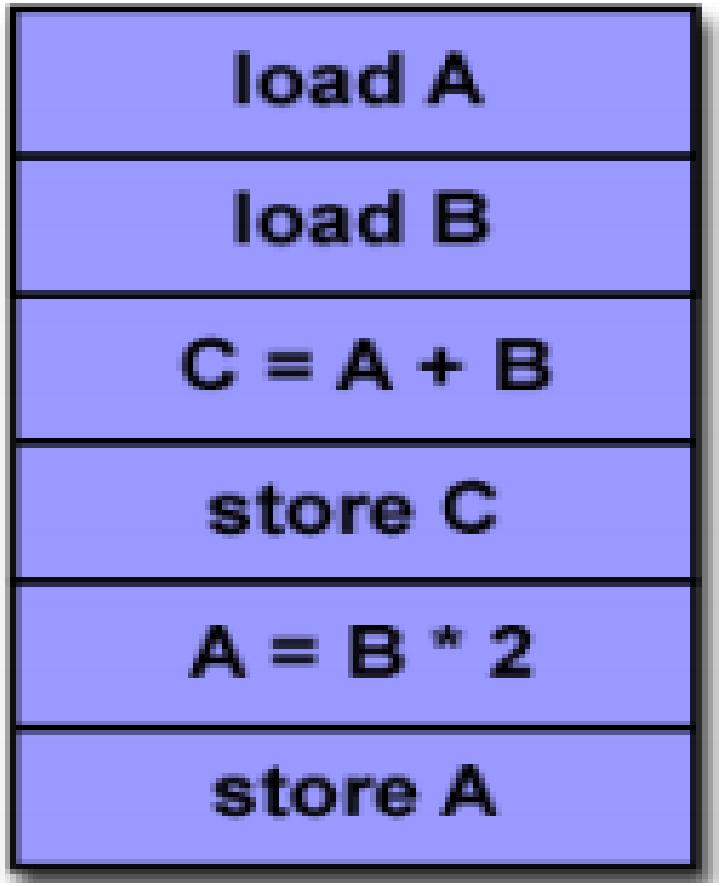
האם יש שאלות מנהליות ?

ארכיטקטורת פון-ניומן (VON NEUMANN)



- ארכיטקטורת פון-ניומן - ארכיטקטורה בסיסית אשר
- פותחה בשנת 1945, הכוללת את ארבעת הרכיבים המרכזיים אשר מתוארים בתרשים.
- כאשר ה- Control Unit מוציא את הפקודות ודואג
- לביצוע שלהן בצורה סיריאלית.

ארכיטקטורת פון-ניומן (VON NEUMANN)



- שיטת העבודה הסיריאלית והבסיסית ביותר.
- בכל מחזור שעון, מועברת למעבד רק פקודה אחת, ומעובד פריט מידע יחיד.

- מהי מקביליות ?
  - "הסדר או מצב המאפשר כמה פעולות או משימות שיש לבצע בו זמנית ולא ברצף"
- במהלך 3 העשורים האחרונים, המקביליות השפיעה כמעט בכל תחום של מדעי מחשב.

•מהו מחשוב מקבילי ?

- מחשוב מקבילי היא טכניקת תיכנות הכוללת "שימוש במעבדים מרובים הפועלים יחד על בעיה אחת".
- הבעיה בסך הכול מחולקת לחלקים, כל אחד מהם מבוצעת על ידי מעבד נפרד ובמקביל.

מה שאנו יכולים לבצע בתוכניות "מקביליות" ?

- מחשבים\מעבדים\ליבות שרצים\ות במקביל.
- רשתות של תחנות עבודה.
- רשתות חישובית (הרכבים ברשת של מחשבים, התקני אחסון, מכשירים מרחוק, וכו') .
- למה לכתוב תוכניות "מקביליות" ?
  - כדי להשיג ביצועים

התכנות המקבילי

- **עקומת הלימוד** של התכנות המקבילי היא מאוד מתונה.
- הסיבה לכך היא מנטלית **ונובעת מהצורך להתרגל לחשיבה מסוג אחר חשיבה מקבילית**. המתכנתים הורגלו לחשיבה – סדרתית, והיא עדיין מוטמעת באופן שבו מתבצעת מלאכת התכנות המקבילי.
- זאת משום **שהזרם המרכזי של התכנות המקבילי מבוסס על שפות תכנות סדרתיות**, שהורחבו לתמוך במקביליות. ולכן, קשה להתנתק מהחשיבה הסדרתית, למרות שהדבר רצוי מאוד.
- לעומת זאת, הקניית חשיבה מקבילית יכולה להיעשות רק בעבודה קשה, דרך אצבעות הידיים. עוד רחוק היום שבו המונח "**תכנות מקבילי**" יהיה שם נרדף למונח "**תכנות**".

האתגרים בתכנות מקבילי

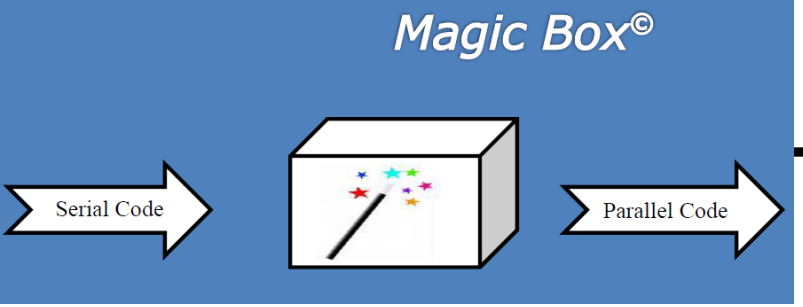
- הפיתוח והמחקר של מערכות תוכנה מקביליות, אינו עומד בקצב המואץ של הפיתוח והמחקר של ארכיטקטורות חומרה מקביליות.
- התכנות המקבילי נחשב לקשה, מייגע, ומועד לתקלים.
- כתיבת יישומים מקביליים יעילים דורשת סביבת פיתוח הולמת, וכלים לניפוי תקלים שיאפשרו תכנות ידידותי נטול טעויות.

חשיבה מקבילית

- תכנות מקבילי דורש מיומנות תכנות חדשות, ואופני חשיבה שונים לפתירת בעיות.
- התכנות הסדרתי הוא תהליך דטרמיניסטי באופיו, ואינטואיטיבי לאופן בו מתכנתים פותרים בעיות באמצעות אלגוריתמים.
- לעומת זאת, התכנות המקבילי הוא לא דטרמיניסטי מטבעו. אלה שהתנסו בתכנות מקבילי מדווחים על קשיים בחשיבה שאיננה לינארית.
- מכול מקום, האי דטרמיניזם הטבוע בתוכניות מקביליות הוא המפתח להשגת - ביצועים גבוהים.
- ניתן לשלוט על ההתנהגות האי דטרמיניסטית - של תוכנית מקבילית, אך רצוי לעשות זאת רק כשהדבר הכרחי, וחשוב לזכור, שניתן ללמוד לחשוב במקביל, ולכתוב תוכניות מקביליות ללא תקלים.

היכרות אינטימית עם החומרה

- תכנות מקבילי דורש מהמתכנת היכרות קרובה עם ארכיטקטורת המחשב עליו הוא עובד.
- היכרות זו, נחוצה לשם התאמת המבנה הלוגי של התוכנית למבנה הפיזי של ארכיטקטורת המחשב, והמעבדים המרכיבים אותו.
- על המתכנת להיות מודע למפרט הבסיסי של המחשב, הכולל פרטים כמו מספר הליבות, המבנה הפיזי של הזיכרון הראשי, והמדרג של זיכרונות מטמון.
- רק כתיבת קוד שלוקחת בחשבון פרטים אלה ואחרים, יכולה להבטיח ביצועים סבירים במערכות מרובות מעבדים וליבות. התכנות הסדרתי חסך מאתנו את הצורך לנבור בקרביים של המחשב.



- רווחה הטענה בקרב הארכיטקטים, שפיתחו את החומרה של המעבדים, שלא נצטרך לתכנת במקביל. הם גרסו, שכול המיקבול יעשה ברמת החומרה, ובאופן שקוף למתכנת, אשר ימשיך לכתוב קוד סידרתי כבימים ימימה.
- הארכיטקטים שפיתחו את התוכנה של המעבדים, היו משוכנעים שהם יוכלו לפתח מהדרים אשר יהיו מסוגלים למקבל קוד סידרתי באופן אוטומטי. גם הם טענו, שאנחנו נמשיך לכתוב קוד סידרתי כבימים ימימה. קוד אשר יועבר כקלט למהדר, שידאג למקבל את הקוד אשר ירוץ על החומרה המקבילית, ויפיק ביצועים מיטביים.
- <u>למרבה הצער, אנו מפתחי התוכנה נותרנו לבדנו, ואין לנו לסמוך אלא על עצמנו</u>.

והמציאות ...

- חשוב שנזכור, שהתכנות המקבילי הוא לא פשוט. לעיתים, נדמה שמדובר בעוד שדרוג פונקציונלי לשפת תכנות קיימת ותו לא או בספריית פונקציות חדשה לתמיכה במקביליות.
- אז זהו שלא. מדובר כאן במהפכת מחשוב של ממש. וכדי להבין את המהפכה היטב, אנו נדרשים ללמוד מחדש את רוב הקורסים שלמדנו משום הארכיטקטורות הן מקביליות, שפות התכנות הן מקביליות. מבני הנתונים הם מקביליים, מסדי הנתונים הם מקביליים, התקשורת היא מקבילית, האלגוריתמים הם מקביליים, הקלט/פלט הוא מקבילי וכן הלאה

הגדרות בסיסיות

1. **מטלה** (Task) - משימה כחלק מעבודה חישובית כוללת. בדר"כ מאופיינת ע"י תוכנית, או סט של פקודות המבוצעות ע"י מעבד.
2. **מטלה מקבילית** (Parallel Task) - מטלה שיכולה להתבצע על מספר מעבדים במקביל.
3. **סנכרון** (Synchronization) - תיאום בין מטלות מקביליות. קביעת נקודות זמן, אשר כל המטלות/מעבדים צריכים להגיע אליה, על מנת להתקדם הלאה.
4. **גרעיניות** (Granularity) - היחס בין כמות החישובים לכמות התקשורת בתוכנית מקבילית. ישנן שתי קטגוריות כלליות:
   1. גרעיניות גסה (Coarse) - כמות גדולה (יחסית) של חישוב מבוצעת בין נקודות תקשורת שונות.
   2. גרעיניות עדינה (Fine) - כמות קטנה (יחסית) של חישוב מבוצעת בין נקודות תקשורת שונות.

הגדרות בסיסיות (המשך)

5. **האצה מקבילית** (Observed Speedup) - הערכה גסה להאצה של תוכנית מקבילית אל מול אותה תוכנית בהרצה טורית (זמן ריצה טורי (סיריאלי) כנגד זמן ריצה מקבילי).
6. **תקורת מקביליות** (Parallel Overhead) - המחיר שיש לשלם (בזמן ריצה) על אופן חישוב\עבודה מקבילי. מאפיינים אשר משפיעים על התקורה הנ"ל הינם לדוגמא: (1) זמני התחלה וסיום של מטלות, (2) סינכרונים, (3) תקשורת מידע בין מטלות\מעבדים, ו-(4)תקורת תוכנה כתוצאה מעיבוד מקבילי.
7. **מקביליות מאסיבית** (Massively Parallel) - שימוש במספר גדול מאוד של מעבדים\מחשבים לעבודה מקבילית.
8. **מקביליות כמעט מוחלטת** (Embarrassingly Parallel)- פתרון של מספר עצום של מטלות דומות מאוד, תו"כ שימוש בתיאום מינימלי ביניהן.

הגדרות בסיסיות (המשך)

9. **יכולת התאמה** (Scalability) - היכולת של מערכת מקבילית להעלות את ההאצה (Speedup) כתוצאה של תוספת מעבדים. דוגמאות לגורמים אשר משפיעים על היכולת הזאת:
   1. חומרה, בפרט תקשורת ורוחב הפס בין המעבד לזיכרון
   2. האלגוריתם שאותן האפליקציה מריצה
   3. תקורת המקביליות
   4. מאפיינים הקשורים לאופן כתיבת האפליקציה (ברמת הקוד, לדוגמא).
10. **אשכול מחשבים** (Cluster Computing) - שימוש במספר מחשבים, ויחידות מסוגים שונים לצורך הרכבת מערכת מקבילית

מחשוב מקבילי - PARALLEL COMPUTERS

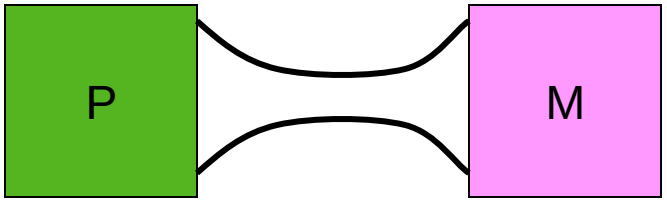
מחשוב מקביל הוא "אוסף גדול של רכיבי עיבוד שיכול לתקשר ולשתף פעולה כדי לפתור את הבעיות גדולות במהירות."

מחשוב מקבילי - PARALLEL COMPUTERS

- "אוסף גדול של רכיבי עיבוד שיכול לתקשר ולשתף פעולה כדי לפתור את הבעיות גדולות במהירות".
- מחשוב במקביל פותח לראשונה, כדי למנוע את צוואר בקבוק "פון-נוימן" (von Neumann bottleneck):
- "זרם ההוראות הוא מטבעו רציף - יש אתר עיבוד אחד, וכל ההוראות, האופרנדים והתוצאות חייבים לזרום דרך צוואר בקבוק בין מעבדים וזיכרון."

צוואר בקבוק VON NEUMANN

- von Neumann bottleneck:



- טכניקות מודרניות שמיועדות ל"להרחיב" את צוואר בקבוק פון נוימן:
  - יחידות פונקציונליות מרובות
  - הקבלה והתקנת צינורות (pipelining) בתוך מעבד
  - מעבד חופף ופעולות I/O
  - זיכרון היררכי.
  - ...

מחשוב מקבילי - PARALLEL COMPUTERS

- "אוסף גדול של רכיבי עיבוד שיכול לתקשר ולשתף פעולה כדי לפתור את הבעיות גדולות במהירות".
- מהו גדול ?
- מעבדים המשמשים במעבדים מקבילים מסיביים (Massively Parallel Processors-MPPs) היכולים להשתנות בכוח והמספר שלהם תלוי בעיצוב הארכיטקטי.
- כלל אצבע: מכונות עם מספר קטן של צמתים (10 צמתים) נוטה להיות בעלי מעבדים חזקים יותר (לדוגמא שרת); מכונות עם מספר גדול מאוד של צמתים (נניח של 10,000 צמתים) נוטה להיות בעלי מעבדים חזקים פחות (לדוגמא כרטיס גרפי).

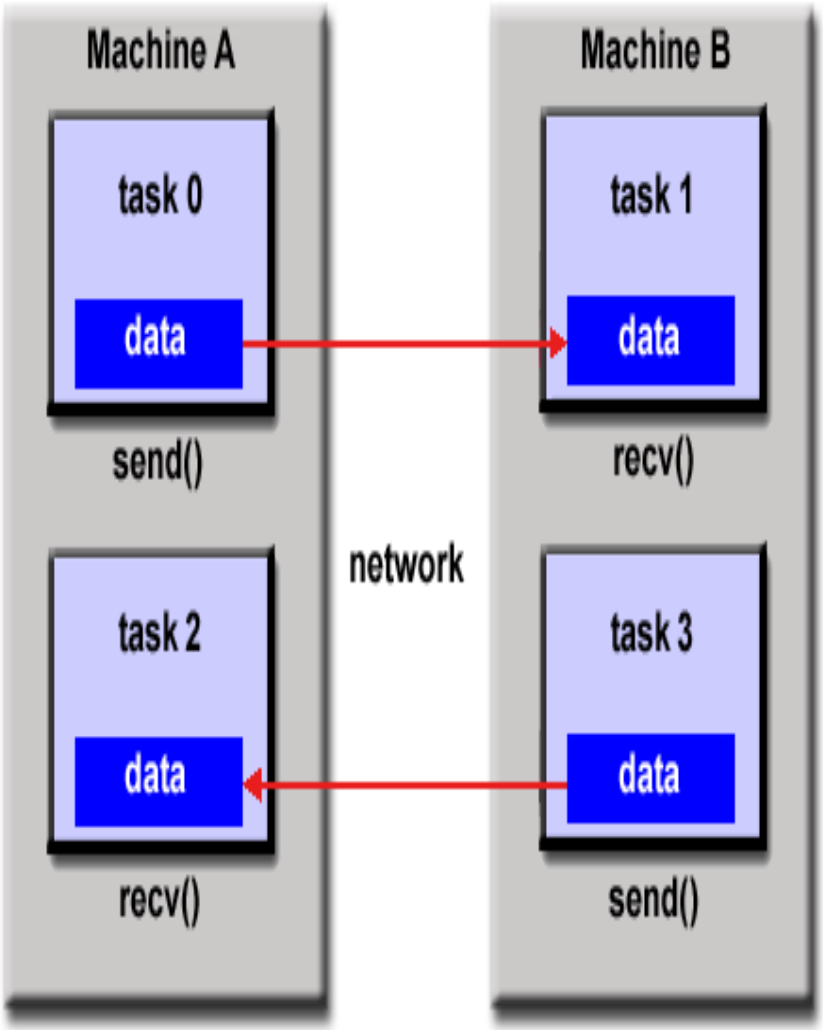
מחשוב מקבילי - PARALLEL COMPUTERS

- "אוסף גדול של רכיבי עיבוד שיכול לתקשר ולשתף פעולה כדי לפתור את הבעיות גדולות במהירות."
- יכולת הרחבה (Scalability):
  - ארכיטקטורה היא מדרגית (*scalable*) אם זה ממשיך להניב את אותם ביצועים פר מעבד כמספר של עליות בכמות המעבדים.
  - מעבדים מקבילים מסיביים (Mpps) מדרגים המתוכננים כך שניתן לבנות גרסאות גדולות יותר של אותו מחשב (כלומר גרסאות עם יותר צמתים\מעבדים) או הרחבות תוך שימוש באותו העיצוב.

מחשוב מקבילי - PARALLEL COMPUTERS

- "אוסף גדול של רכיבי עיבוד שיכול לתקשר ולשתף פעולה כדי לפתור את הבעיות גדולות במהירות."
- מעבדים חייבים לתקשר אחד עם השני ועם העולם החיצון.
- שתי פרדיגמות תקשורת MPP סטנדרטיות:
  - העברת מסרים (message passing): מעבדים מתקשרים באמצעות שליחת הודעות אחד לשני
  - זיכרון משותף (Shared memory): מעבדים מתקשרים על ידי גישה למשתנים משותפים

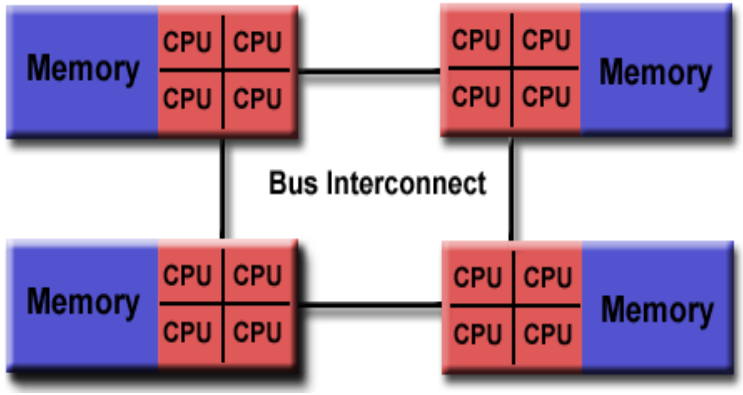
מודל העברת מסרים (MPI)



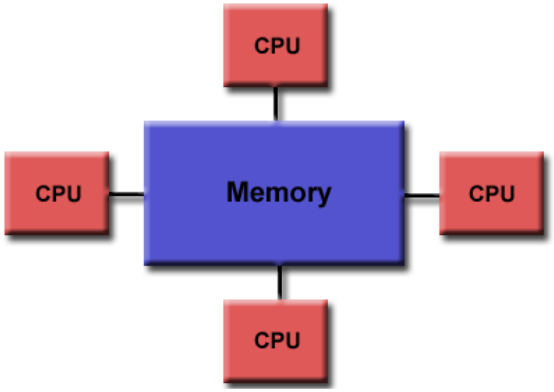
- סט של מטלות, אשר לכל אחת זיכרון לוקאלי משלה. מטלות אינן מוגבלות למחשב יחיד, ויכולות להתפרס על מספר כלשהו של מחשבים/אשכולות.
- המטלות מעבירות מידע בעזרת תקשורת המבוטאת בשליחת וקבלת הודעות.
- בדר"כ, העברת מידע הינה פעולה מתואמת, לדוגמא, לכל שליחה מצד אחד, תהיה קבלה בצד השני.
- דוגמא למודל הנ"ל, הינו מנגנון ה- *MPI*.

זיכרון משותף (SHARED MEMORY)

- המטרה היא לאפשר למספר המעבדים השונים לגשת לאותו זיכרון ולפעול עליו.
- בצורה זו, מעבדים יכולים לשתף ביניהם מידע רלוונטי, ושינויים של מעבד כלשהו זמינים למעבדים האחרים.

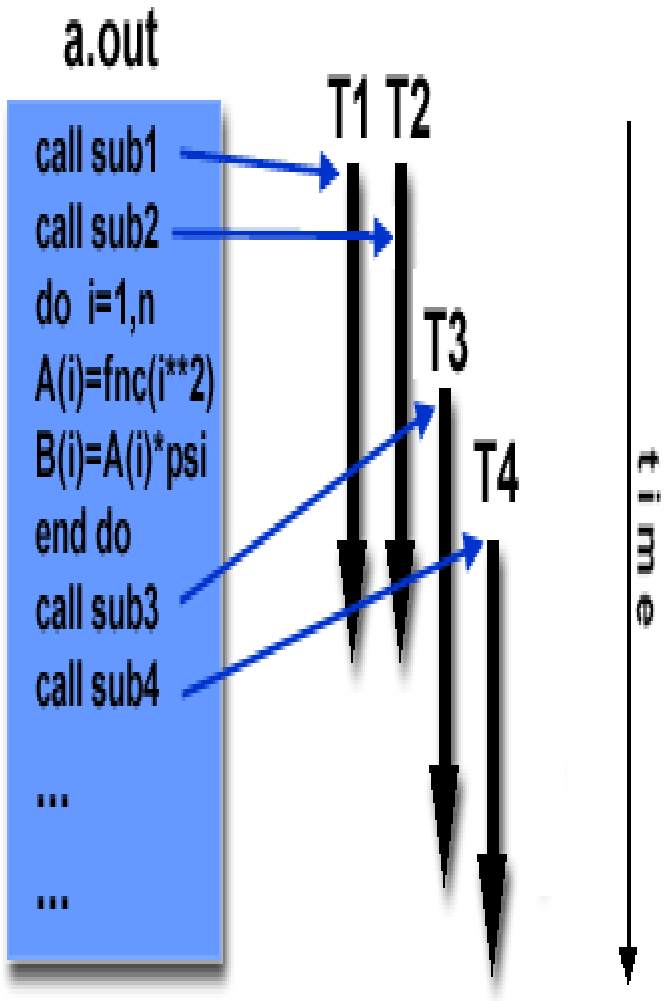


Non-Uniform Memory Access - זמני הגישה לזיכרון בין המעבדים השונים אינם זהים. הגישה לזיכרון דרך הקישור של ה-Bus היא איטית יותר.



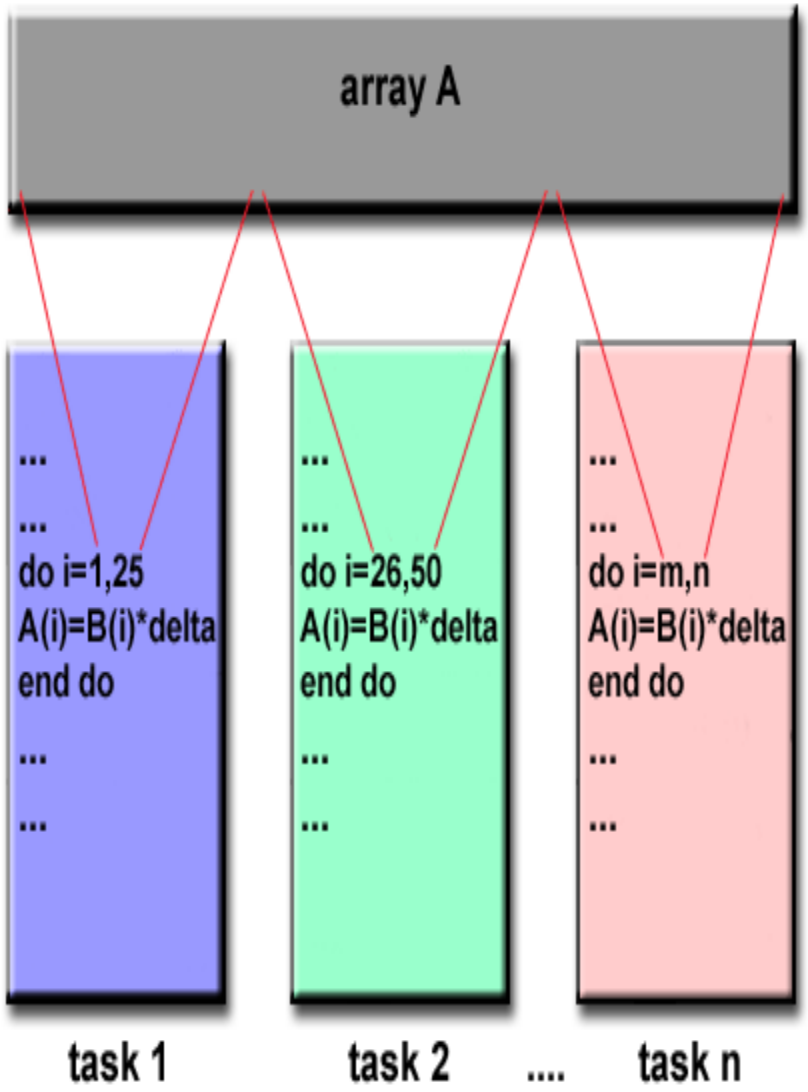
Uniform Memory Access - המעבדים זהים, וכל אחד מהם יכול לגשת לזיכרון המשותף.

מודלים נוספים למחשוב מקבילי



- **מודל החוטים** (Threads) - לתהליך מסוים יכולים להיות מספר מסלולי חישוב בו-זמנית. כאשר תהליך מתחיל לרוץ, הוא יכול ליצור חוטים, אשר משתפים ביניהם זיכרון, ובצורה זו לפצל את העבודה שלו, למספר תתי-עבודות, אשר בדר"כ יכולות לרוץ במקביל.
- לכל חוט ישנה "ראיה גלובלית" של הזיכרון. החוטים מתקשרים בעזרת הזיכרון המשותף

מודלים נוספים למחשוב מקבילי



- **מודל המידע המקבילי (מבוזר)** - במודל הנ"ל מבוצעת עבודה על אוסף של מידע, אשר מאורגן במבנה נתונים ידוע מראש (אשר מאפשר גישה מקבילית/מסודרת).
- קבוצה של מטלות פועלות על אותו **מבנה הנתונים**, אך כל מטלה (פועלת\אחראית) על חלק אחר מהמבנה. ובנוסף, כל המטלות מבצעות את אותה הפעולה.
- **מבנה הנתונים** מחולק לחתיכות (חלקי מידע) וכל מטלה מקבלת אחריות על כמות מסוימת של חתיכות (חלקי מידע).

מחשוב מקבילי - PARALLEL COMPUTERS

- "אוסף גדול של רכיבי עיבוד שיכול לתקשר ולשתף פעולה כדי לפתור את הבעיות גדולות במהירות."
- כמה זמן זה לוקח כדי לתקשר ?
- מדדי רשת רלוונטיים:
  - רוחב פס (Bandwidth): מספר הביטים לשנייה שיכולים להיות מועברים באמצעות הרשת.
  - השהייה (Latency): זמן לוקחת העברת המסר דרך הרשת.

מחשוב מקבילי - PARALLEL COMPUTERS

- "אוסף גדול של רכיבי עיבוד שיכול לתקשר ולשתף פעולה כדי לפתור את הבעיות גדולות במהירות."
- תוכניות להעברת מסרים מקבילית, יכולות למזער עיכובי תקשורת על ידי חלוקת התכנית לתהליכים ולהתחשב בגרעיניות (granularity) של התהליך במחשב.

$$granularity = \frac{t_{computation-\text{חישוב}}}{t_{communication-\text{תקשורת}}}$$

גרעיניות (Granularity) - היחס בין כמות החישוב לכמות התקשורת בתוכנית מקבילית. בדר"כ, שלבי חישוב מופרדים משלבי התקשורת בעזרת פעולות\ארועי סנכרון.

מחשוב מקבילי - PARALLEL COMPUTERS

- "אוסף גדול של רכיבי עיבוד שיכול לתקשר ולשתף פעולה כדי לפתור את הבעיות גדולות במהירות."
- תוכניות מורכבות מתהליכים\משימות אשר עשוי להיות תלויים זה בזה.
  - הארכיטקטורה\התוכנה חייבת לספק תמיכה לסנכרון (synchronization).
- "סנכרון גבול" (barrier synch) משמש בדרך כלל לתיאום תהליכים.
- תכנית עם תהליכים עצמאיים שנקראה "מקביליות כמעט מוחלטת - embarrassingly parallel" (פתרון של מספר עצום של מטלות דומות מאוד, תו"כ שימוש בתיאום מינימלי ביניהן).

סנכרון (SYNCHRONIZATION)

- "סנכרון גבול" או "מחסום גבול" (synchronization Barrier) - נקודה בעת הביצוע, אשר אליה צריכות להגיע כל המטלות המעורבות בפעולה.
- כל מטלה מבצעת את קטע הקוד שלה, עד שהיא מגיעה למחסום, ואז נאלצת לחכות עד שכל המטלות האחרות יגיעו למחסום (נחסמת/נעצרת).
- כאשר המטלה האחרונה מגיעה למחסום, כל המטלות הינן מסונכרנות.

מחשוב מקבילי - PARALLEL COMPUTERS

- "אוסף גדול של רכיבי עיבוד שיכול לתקשר ולשתף פעולה כדי לפתור את **הבעיות גדולות** במהירות."
- מהו גדול ?
- ניתן להשתמש ב-MPPs (massively parallel processor array)

  לפתור בעיות ש:
  - לא ניתן לפתור במסגרת זמן סבירה.
  - לא ניתן לפתור בגדלים שבהם יש עניין.
  - לא ניתן לפתור בזמן אמת.
  - אינו כדאי מבחינה כלכלית (כ"א, זמן,..).

  עם מעבד יחיד

מחשוב מקבילי - PARALLEL COMPUTERS

- "אוסף גדול של רכיבי עיבוד שיכול לתקשר ולשתף פעולה כדי לפתור את הבעיות גדולות במהירות."
- מה מהיר\טוב ?
- תלוי איך אתה מודד את ביצועים

$$Speedup = \frac{t_{serial-\text{סידרתי}}}{t_{parallel-\text{מקבילי}}}$$

$$giga = 10^9$$

$$tera = 10^{12}$$

$$peta = 10^{15}$$

מגבלות המקביליות - חוק אמדל

- החסם עליון על הפוטנציאל להאצה מאופיין ע"י גודל קטע הקוד P אשר ניתן למיקבול הינו:

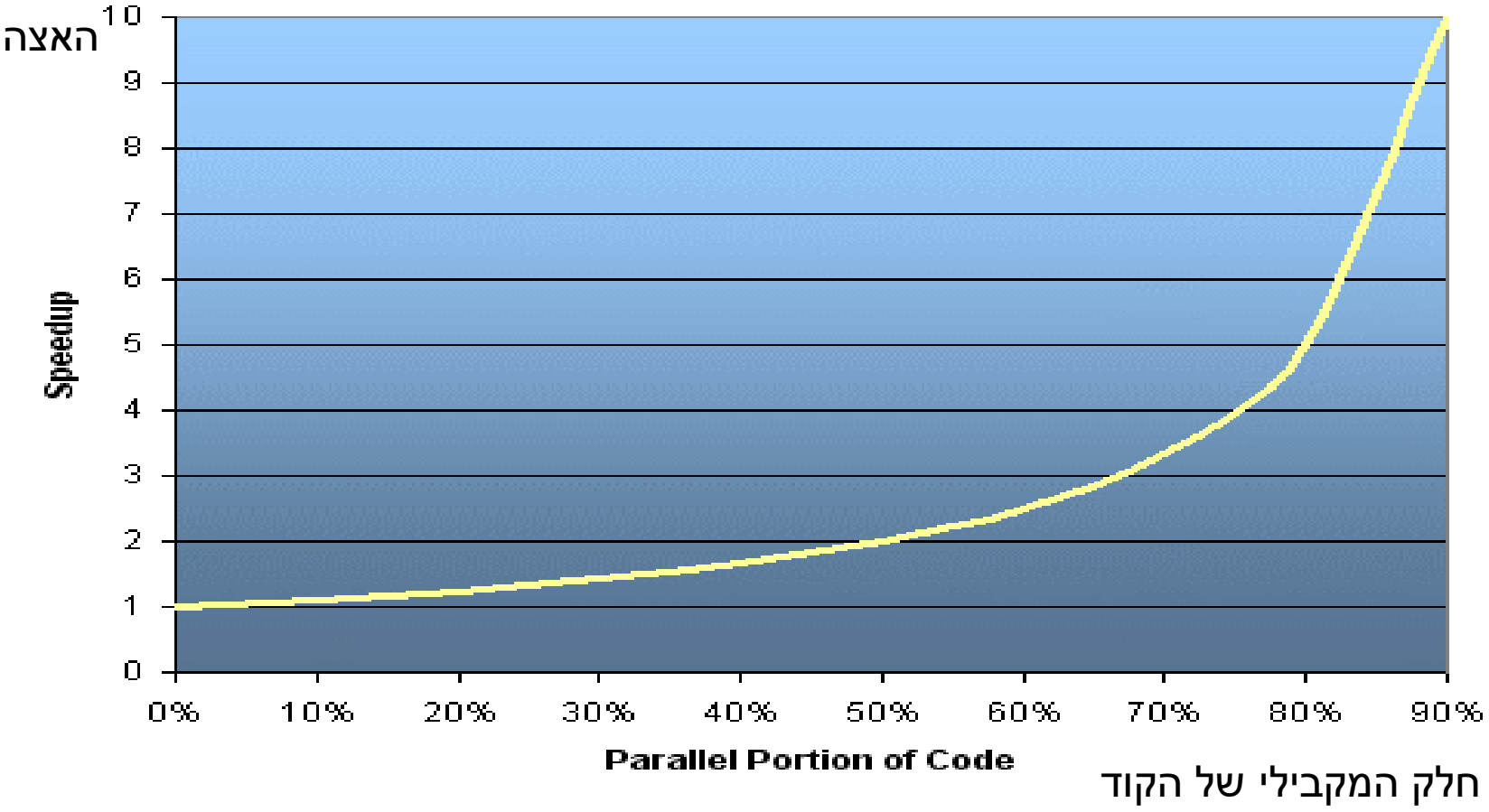
$$\frac{1}{1-P}$$

- כאשר נתון מספר המעבדים אשר מבצעים את קטע הקוד שניתן להאצה P (נסמן אותם ב-N), נקבל כי הפוטנציאל להאצה הינו:

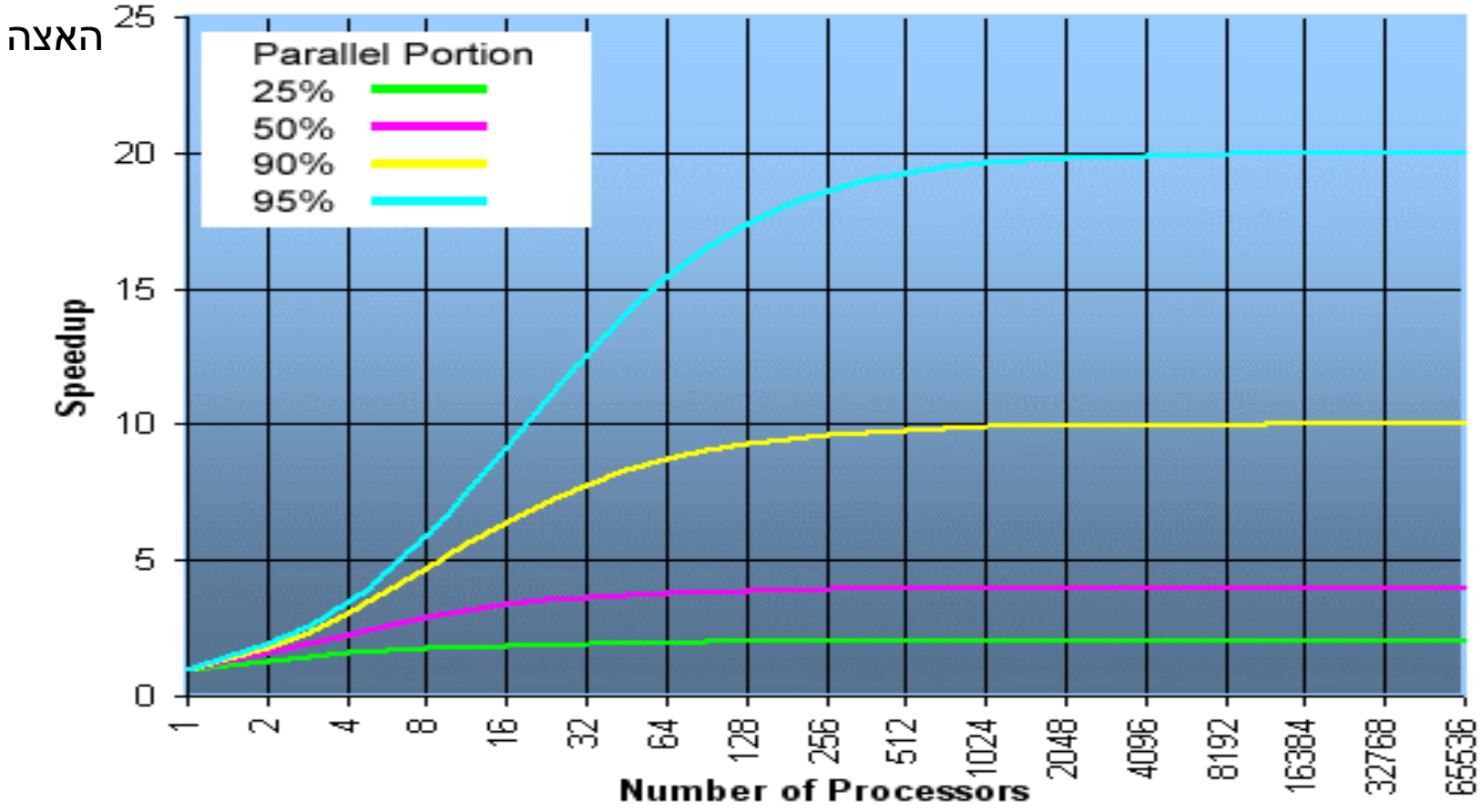
$$\frac{1}{\frac{P}{N}+S}$$

- כאשר S הינו גודל קטע הקוד אשר נשאר סריאלי.

מגבלות המקביליות - חוק אמדל



מגבלות המקביליות - חוק אמדל



מגבלות המקביליות - סיבוכיות

- בדר"כ, תוכניות מקביליות מסובכות יותר מתוכניות סיריאליות בכמה סדרי גודל.
- מכיוון שלא רק שפעולות מבוצעות במספר ערוצים, ישנו גם מידע אשר מועבר ביניהן.

מגבלות המקביליות - ניידות

- מערכות ההפעלה הן אלו שמחליטות על מדיניות הזימון ואופן ההפעלה המקבילי, ולכן, לא ניתן להניח דבר, ולהסתמך על מאורעות\תוצאות במערכת הפעלה אחת, כאשר עובדים עם מערכת הפעלה אחרת.
- ארכיטקטורת החומרה והתמיכה במקביליות ברמה החומרה עלולה גם היא להשפיע על הניידות.

מגבלות המקביליות - דרישות משאבים

- המטרה של מקביליות היא להאיץ את זמן הריצה של תוכנית כלשהי, הדבר גורר שימוש מוגבר ב-CPU ויכול להשפיע לרעה על צריכת המשאבים.
- בנוסף, בדר"כ, דרישות הזיכרון של תכנות מקבילי הינן, גדולות יותר מאשר תכנות סיריאלי, ושכפולי המידע הנדרשים יכולים להשפיע על זמן הריצה.
- למעשה, יכול להיות כי עבור תוכניות מקביליות אשר פועלות זמן קצר, ריצה סיריאלית על אותה התוכנית היה מהירה יותר, עקב התקורה ביצירת סביבת ריצה מקבילית, אתחולים מתאימים והתקשורת הנדרשת.

מגבלות המקביליות - יכולת התאמה

- לפעמים, הוספה של מכונות\מעבדים\משאבי חישוב אינה מקצרת את זמן החישוב, אלא רק מגדילה אותו, וזאת, כתוצאה מהתקורה הקיימת בתקשורת\גישה למידע וכו'.
- בנוסף, נכנסות כאן מגבלות חומרה, כדוגמת גודל רוחב הפס בין הזיכרון למעבדים, רוחב הפס של התקשורת בין המטלות, גודל הזיכרון אשר ניתן לכל מכונה, זמן המחזור וכו'.
- נוסף לאלו, ספריות אשר תומכות במקביליות (לדוגמא POSIX, MPI OpenMP ועוד) יכולות בעצמן ליצור מגבלות ללא קשר לאפליקציה.

10 מחשבי העל המהירים ביותר

1. ASCI Red [Intel, 9632 processors)
2. ASCI Blue-Pacific [IBM SP, 5808 processors]
3. ASCI Blue-Mountain [SGI, 6144 processors]
4. NAVOCEANO SP [IBM, 1336 processors]
5. SR800-F1/112 [Hitachi, 112 processors]
6. SR800-F1/112 [Hitachi, 100 processors]
7. T3E1200 [Cray, 1084 processors]
8. T3E1200 [Cray, 1084 processors]
9. SR800/128 [Hitachi, 128 processors]
10. T3E900 [Cray, 1324 processors]
16. Blue Horizon [IBM, 1152 processors]

מה אתם יכולים לקנות היום ?

- IBM SP [e.g. Blue Horizon]
- SGI Origin 2000 [distributed shared memory]
- Sun HPC 10000 [SMP front-end and SMP compute engine]
- Compaq Alpha Cluster
- Tera MTA [Multithreaded architecture]
- Cray SV-1 Vector Processor
- Fujitsu and Hitachi Vector Supers

LOCAL COMPUTER MAKES GOOD: BLUE HORIZON @ SDSC -- WORLD'S 16TH FASTEST MACHINE (AS OF JUNE, 2000)

מגמה נוכחית בארכיטקטורות מחשוב מקבילית: אשכולות (CLUSTERS)

- מעין מחשב על עלוב (Poor man's Supercomputer)
- ערימה של מחשבים (A pile-of-PC's).
- שימוש א'טרנט (Ethernet) או ברשת במהירות גבוהה (Myrinet)
- שימוש בארכיטקטורת חומרה גבוה (בעתיד הקרוב).
- בעיקרו של דבר בניה עצמית של מעבדים מקבילים (MPP).