Computer in der Wissenschaft

Verschiedene Anwendungen

- Numerische:
  - Simulationen von Materialien oder Detektoren
  - Berechnungen in der Gitter-Quantenchromodynamik
- Symbolische/Analytische:
  - Berechnungen von Feynmandiagrammen in der Quantenfeldtheorie
  - Berechnung von Wickkontraktionen für Korrelationsfunktionen

Einige aktuelle Forschungsbeispiele findet man hier:
`https://indico.cern.ch/category/7679/`
`https://indico.cern.ch/event/567550/`
Advanced Computing and Analysis Techniques

Unterschiedliche Plattformen:

- Desktop Systeme/Laptops
- Manycore Server
- GPU (FPGA)
- HPC Cluster

Unterschiedliche Sprachen

- Numerik:
  - Hochsprachen C(++), Fortran, Python
  - Assembler
- Symbolisch
  - Mathematica
  - FORM

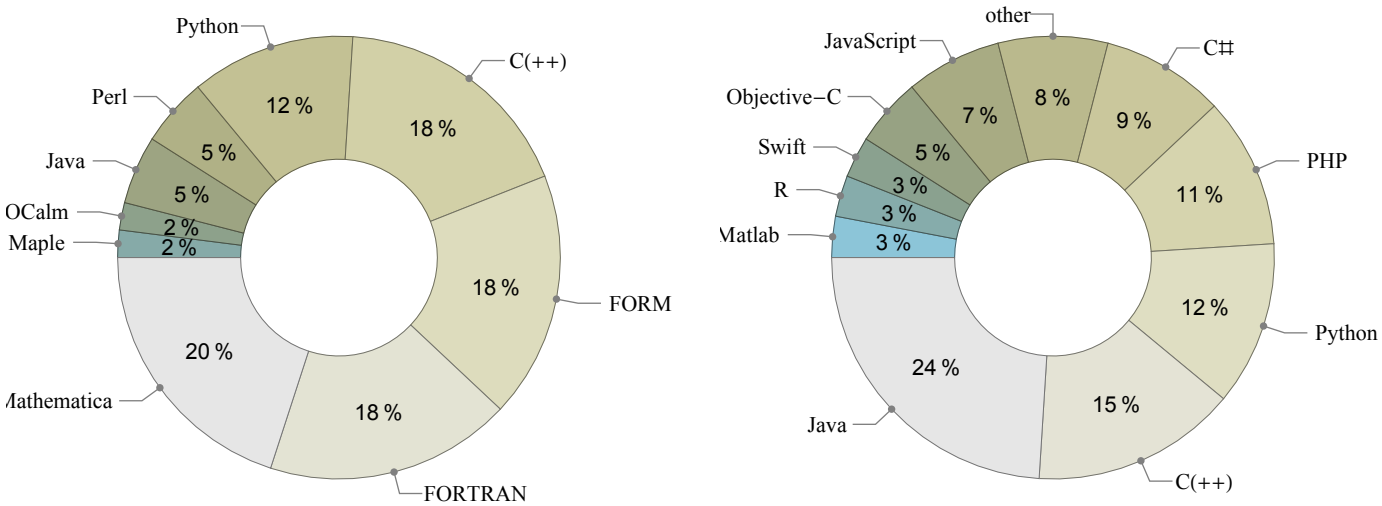

Figure 1. The distribution of programming languages and tools used in the theoretical high-energy physics community (left panel) and in the software industry (right panel). For the former, we have analyzed the packages presented and used by the participants of the session. For the latter, data from the PYPL index was used [3].

G. Luisoni, S. Poslavsky and Y. Schroder, J. Phys. Conf. Ser. **762**, no. 1, 012077 (2016) doi:10.1088/1742-6596/762/1/012077 [arXiv:1604.03370 [hep-ph]].

Pros und Cons

Hochsprachen:

Vorteile:

- Time to Solution ist sehr kurz - Schnellere Programme
- Hohes Maß an Kontrolle
- Auf den meisten Systemen (frei) verfügbar
- Zahlreiche (frei) verfügbare *Bibliotheken*
- Lineare Programmausführung (meistens)

Nachteile:

- Flache Lernkurve
- (teilweise viel) Eigenentwicklung nötig
- Eingeschränktes Userinterface
- I/O umständlich
- (wenig) symbolische Fähigkeiten

Pros und Cons

Mathematica:
Vorteile:

- Time to Application ist sehr kurz - Rapid Development
- Intuitives Interface (Notebooks)
- Symbolische Manipulationen
- Umfangreiche Bibliotheken zur analytischen und numerischen Berechnung
- Umfangreiche Dokumentation
- Viele (freie) Zusatzpakete

Nachteile:

- Langsam (teilweise sehr)
- Proprietär (und teuer)
- In Teilen eine Black Box

Was soll man nehmen?

- Häufig arbeitet man mit mehreren Systemen, je nach eigener Präferenz.
- Mathematica ist vielfältig einsetzbar:
  - Es gibt Schnittstellen zu externen Programmen in C (WSTP früher Mathlink)
  - Man kann Mathematica nutzen um Programme in einer anderen Sprache zu schreiben.
  - Die Visualisierung ist recht einfach in Mathematica realisierbar
  - Das Notebook-Interface ist eine Stärke, da die Programme meist optisch nah an den mathematischen Formeln sind
- Ich benutze häufig:
  C, Python, FORM und Mathematica

Packages

Mathematica ist erweiterbar in Form von Paketen.
Einige Beispiele für Mathematica Pakete in der Hochenergiephysik sind:

- FeynCalc - weit verbreitet zur Berechnung von Feynman Diagrammen
- FeynArts - malt alle Diagramme zu einer gegebenen Topologie in einem definierten Modell
- LoopTools - numerische Berechnung von Einschleifenintegralen (Beispiel einer mathlink Anwendung)

Datenformate

Daten müssen für Computer passend *kodiert* werden.
Letztlich sind alles nur bits . . .
Leider nicht!
Little und Big Endian
In Welcher *Reihenfolge* werden die Bitmuster gespeichert?
Zwei Möglichkeiten:

- Höchste Stelle steht am Anfang (Big Endian)
- Höchste Stelle steht am Ende (Little Endian)

Zahlen

Darstellung von Integerzahlen

- Einfachste Darstellung von vorzeichenlosen Integerwerten über Binärdarstellung.

Beispielhaft für m bits.

$$\underbrace{00\ldots0}_{m} = 0$$

$$\underbrace{00\ldots1}_{m} = 1$$

$$\underbrace{11\ldots1}_{m} = 2^m - 1 \tag{1}$$

bei m bits können wir von 0 bis $2^m - 1$ codiren. Das reicht noch nicht zur eindeutigen Festlegung

$$\underbrace{10\ldots00}_{\text{Little Endian}} = 1 \ \vee \ \underbrace{00\ldots01}_{\text{Big Endian}} = 1 \tag{2}$$

Byteorder kann ein Problem sein wenn man auf verschiedenen Architekturen läuft, z.B. i.e. BlueGene vs x86_64.

Darstellung vorzeichenbehafteter Integer

Mehrere Möglichkeiten

Sign/Magnitude

Natürlichste Methode mit führendem Vorzeichenbit + Absolutwert. Bei m bits ergeben sich als darstellbare Zahlen

$$\left[-2^{m-1}+1, 2^{m-1}-1\right] \tag{3}$$

Probleme sind hier technischer Natur (Rechenlogik muss mit dem Extrabit zurechtkommen) und es gibt die Null zwei mal.

Zweierkomplement

Alternative als Darstellung im Zweierkomplement

- positive Zahlen (inkl. 0) starten mit 0 im führenden Bit

Konstruktion der negativen Zahl $a < 0$

1. Schreib $|a|$ in Bit-notation
2. Negiere das Ergebnis
3. Addiere eins

Beispiel -5 in Zweierkomplementdarstellung

$$\begin{aligned} (5)_2 &= 0b0101 & & \\ & \quad 0b1010 & & |\text{negate} \\ (-5)_2 &= 0b1011 & & |+1 \end{aligned} \tag{4}$$

Vorteile sind einfachere Implementierung in Hardware.

Biased-Darstellung

Man definiert einen konstanten offset (Bias) um den man die Werte verschiebt:

$$\bar{x} = x + b \tag{5}$$

m sei die Bitanzahl und damit der darzustellenden Wertebereich $[-2^{m-1} + 1, 2^{m-1} - 1]$.
Verschiebt man diesen nun um den Bias $b = 2^{m-1} - 1$ so sind alle Zahlen positiv.
Nachteile liegen in der komplexeren Implementierung von Rechenoperationen.

Gleitkommadarstellung (Fix-/Festkommadarstellung)

Darstellung der reellen Zahlen etwas komplizierter

- Sign: Erstes bit
- Exponent: 8 (11) bits (power of 2), in Biased 127 (1023) Darstellung
- Mantissa: Rest 23 (52) bits

wobei die Zahl in Klammern für double precision steht.
Jede reelle Zahl n ist dann gegeben durch

$$n = s \times Base^{e-B} \times Mantisse \tag{6}$$

Frage nach der Defintion der Mantisse.

Normalisierte Zahlen

Frage:
Wie soll z.B. die Zahl zwei dargestellt werden (dezimal)?

$$2 = 2 \cdot 10^0$$
$$2 = 0.2 \cdot 10^1 \tag{7}$$

Häufige Definition ist das die führende Stelle zwischen
$1 \leq n < b$ liegt wobei b die Basis ist.
Wenn b=2 ist das immer 1 und kann bei der Kodierung der Zahl weggelassen werden.
Das Format ist dann

$$n = s \times 2^{e-B} \times 1.m \tag{8}$$

Beispiel

Die Zahl 0.2 als floating point single precision nach IEEE.

- Das Vorzeichenbit ist 0.
- Der Exponent der normalisierten Zahl ist 2^{-3}, denn $0.2 = 1.6 \times 2^{-3}$ *Rightarrow* in Biasdarstellung 124.
- Ohne die führende 1 müssen wir 0.6 als Binärzahl schreiben

$$
\begin{aligned}
.6/2^{-1} &\to 1 \\
.1/2^{-2} &\to 0 \\
.1/2^{-3} &\to 0 \\
.1/2^{-4} &\to 1 \\
.0375/2^{-5} &\to 1
\end{aligned}
\tag{9}
$$

Insgesamt

$$(0.2) = \underbrace{0}_{+} | \underbrace{01111100}_{124} | \underbrace{10011001100110011001101}_{.600000023841858} \quad (10)$$

das ergbit

$$0.2 = 0.200000002980232238769531250 \quad (11)$$

Der Abstand zwischen zwei Zahlen ist nicht konstant, insbesondere gibt es eine große Lücke in der Nähe der Null. Ausweg denormalisierte Zahlen.

- Alle Epxonentenbits sind 0 $E \Rightarrow 2^{-126}$
- Interpretiere die führende Stelle als Null

$$n = s \times 2^{-126} \times 0.m \tag{12}$$

Kleinste Zahl ist für single precision

$$n = s \times 2^{-126} \times 2^{-23} = s \times 10^{-149}$$

in normalisierter Schreibweise

$$n = s \times 2^{-126} 1.0 = 10^{-126}$$

Rundungsfehler

Frage:
Was ist der maximale relative Rundungsfehler einer IEEE Floating point Zahl?

$$\frac{|x - \mathrm{rd}(x)|}{|x|} = \epsilon \tag{13}$$

Zwei benachbarte Zahlen haben den Abstand

$$b^{-m} \tag{14}$$

wobei b die Basis und m die Mantissenlänge ohne führende Stelle. Damit ist der maximale relative Fehler

$$\epsilon \leq b^{-m-1} \tag{15}$$

Gilt nur für normalisierte Zahlen.

Andere Datenformate

Es gibt noch viele weitere Datenformate (ASCII, Bitmap, Postscript, usw.)
In der Physik macht es häufig Sinn weitere Informationen zu speichern:

- Was genau bedeuten die Daten, z.B. Messung, Fehler in Spalten?
- Wie kann man auf die Daten zugreifen?
- Solche Informationen nennt man auch Metadaten

Serh sinnvoll: Daten + Metadata in einem, z.B. als Notebook.
Andere verbreitete Formate sind HDF5, XML