La performance des plates-formes Java Card

J. Cordry

CNAM-Cédric

2 Juin 2008

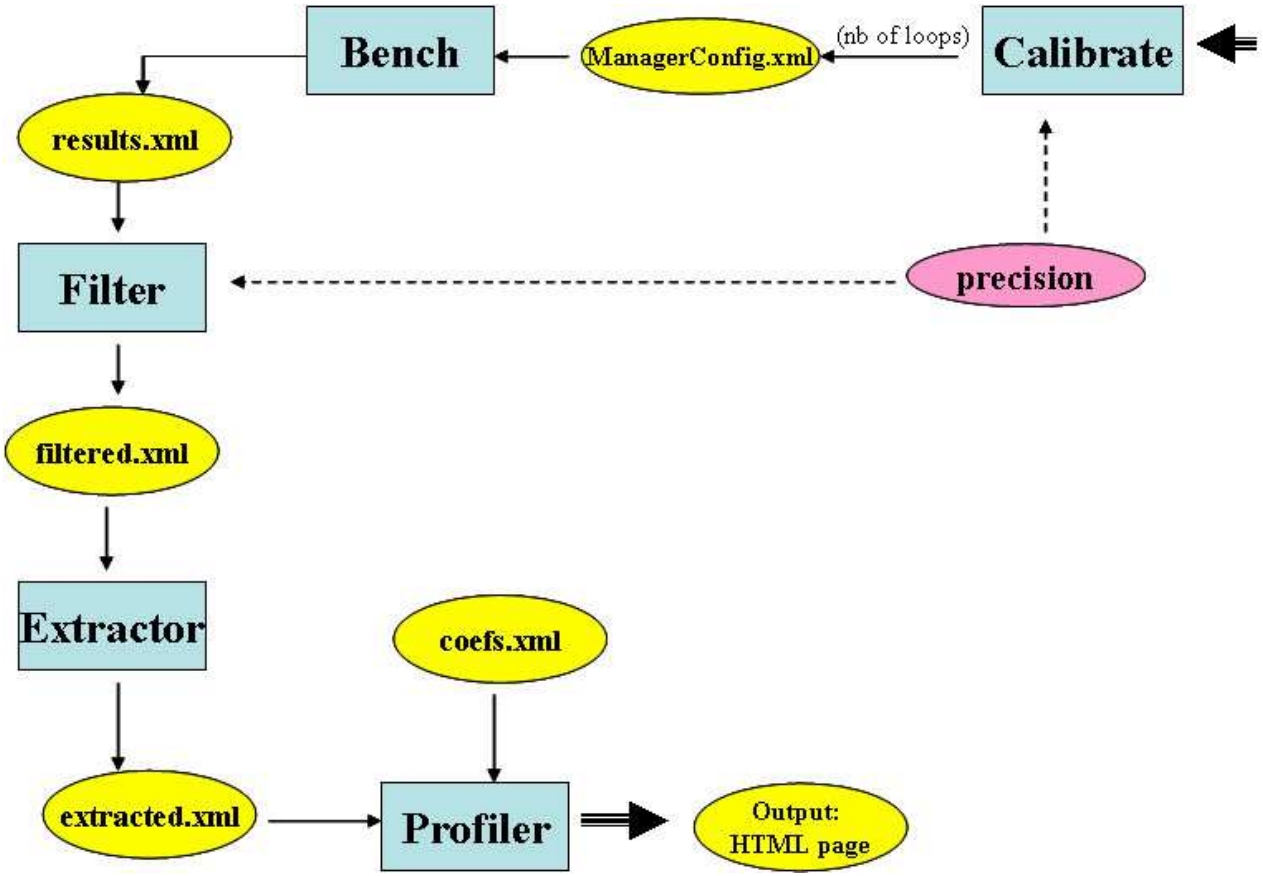
Le projet Mesure

- Le projet MESURE a été financé par l'ANR RNTL - sélectionné en 2005
- Durée : Mai 2006 a Mars 2008
  `http://mesure.gforge.inria.fr/`
- Partenaires :
  - CNAM-Cédric (Paris)
  - INRIA POPS (Université de Lille),
  - Trusted Labs (Sophia Antipolis).

Java Card

- Une architecture client/serveur (la carte est le serveur)
- Une machine virtuelle embarquée sur carte à puce
- Mono-tâche (jusqu'à très récemment)
- Sous-ensemble très réduit du langage Java
- Les classes compilées doivent être converties
- Le programme généré (applet) peut ensuite être chargé sur une carte
- Les développeurs doivent tenir compte des capacités très réduites des cartes à puces (mémoire/CPU)

Framework


Bench
ManagerConfig.xml
(nb of loops)
Calibrate
results.xml
Filter
precision
filtered.xml
Extractor
coefs.xml
extracted.xml
Profiler
Output:
HTML page

Ce que l'on veut mesurer

- On démarre un chronomètre en lançant un APDU "run" à l'applet de test
- On arrête le chronomètre quand la carte réponds un APDU.
- On effectue la mesure plusieurs fois (un nombre Y)
- L'applet fait tourner le cas de test un certain nombre de fois ($X = P2 \times P2$)

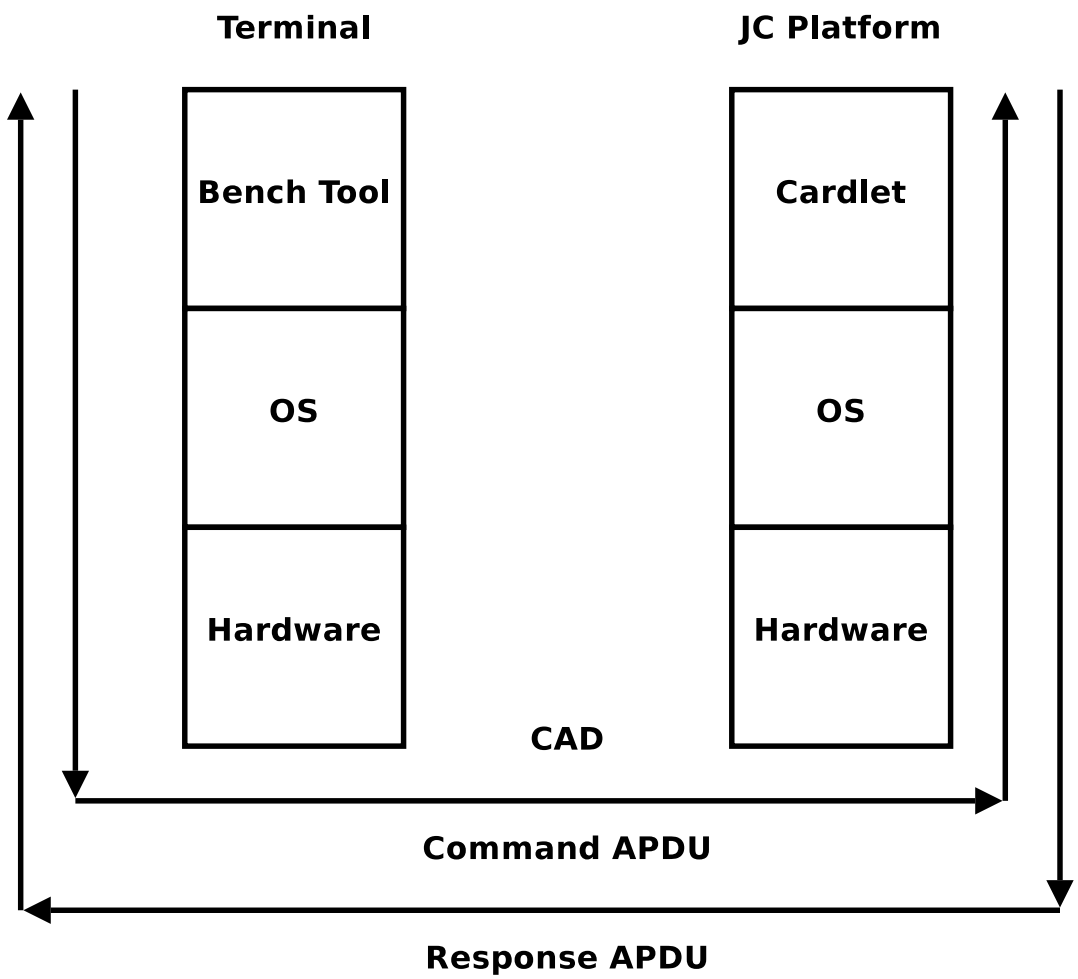
Bruit des mesures

Le mesures expérimentales sont sujettes à des erreurs dues au bruit dans:

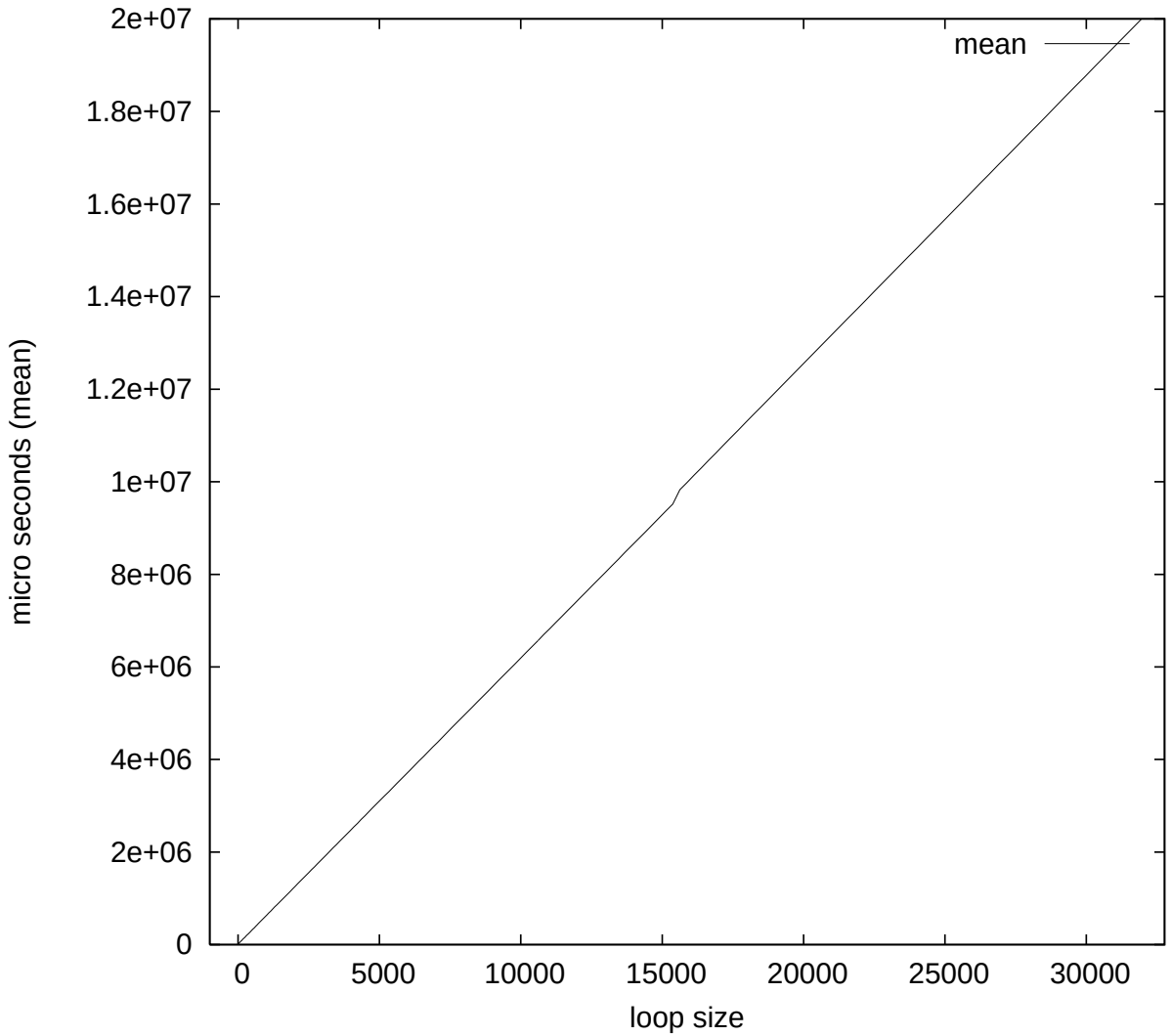
- La plate-forme Java Card elle-même et le lecteur
- Le terminal (la machine qui sert a faire les tests : outil de mesure/JVM/OS/hardware)

Faire des mesures prends du temps

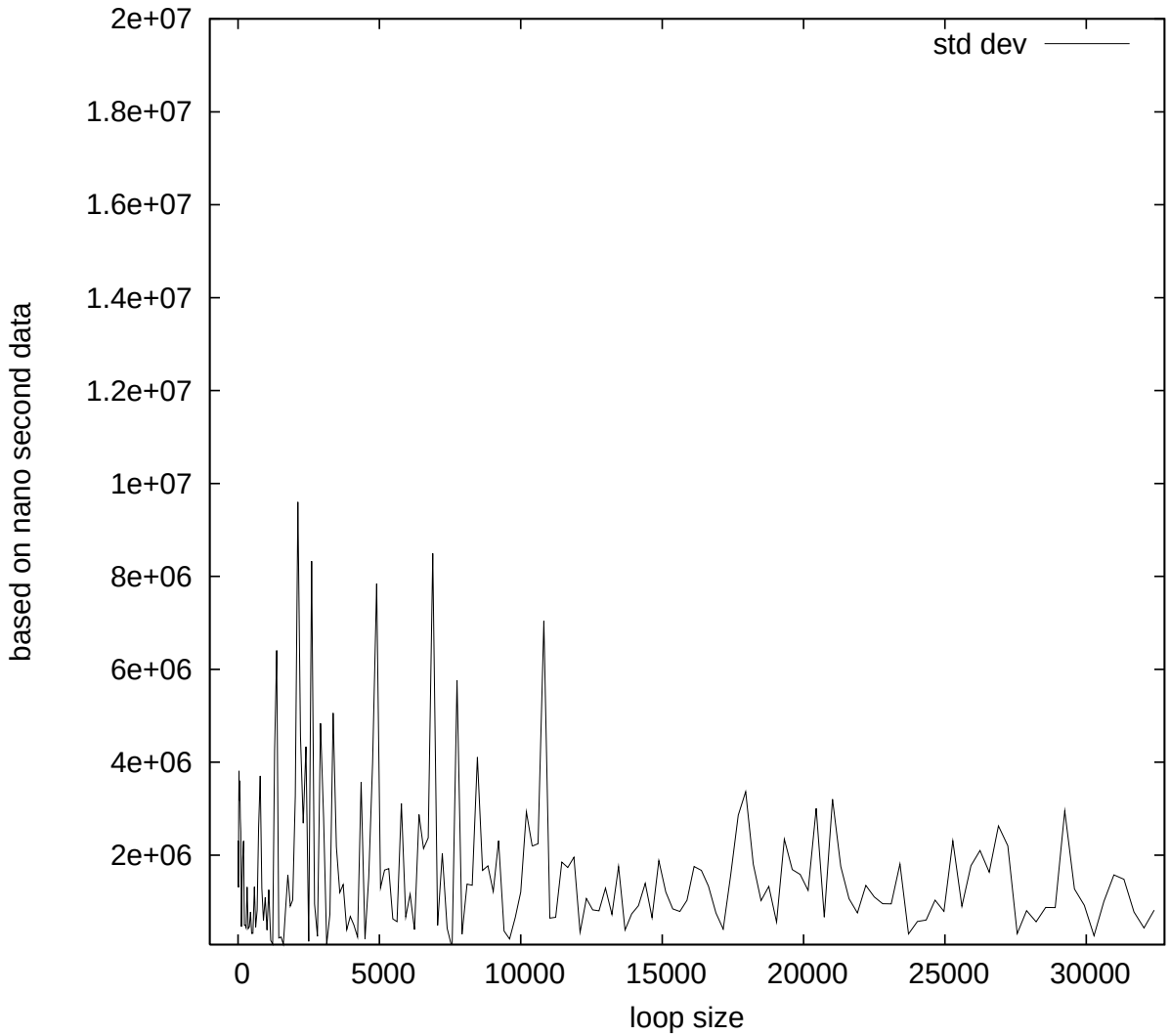
- On veut mesurer le temps de performance avec une boucle sur la carte et sur la machine/terminal de test
- Une grande taille de boucle tends à minimiser les bruits de mesure
- On ne veut pas passer trop de temps à mesurer


Terminal
JC Platform
Bench Tool
OS
Hardware
Cardlet
OS
Hardware
CAD
Command APDU
Response APDU

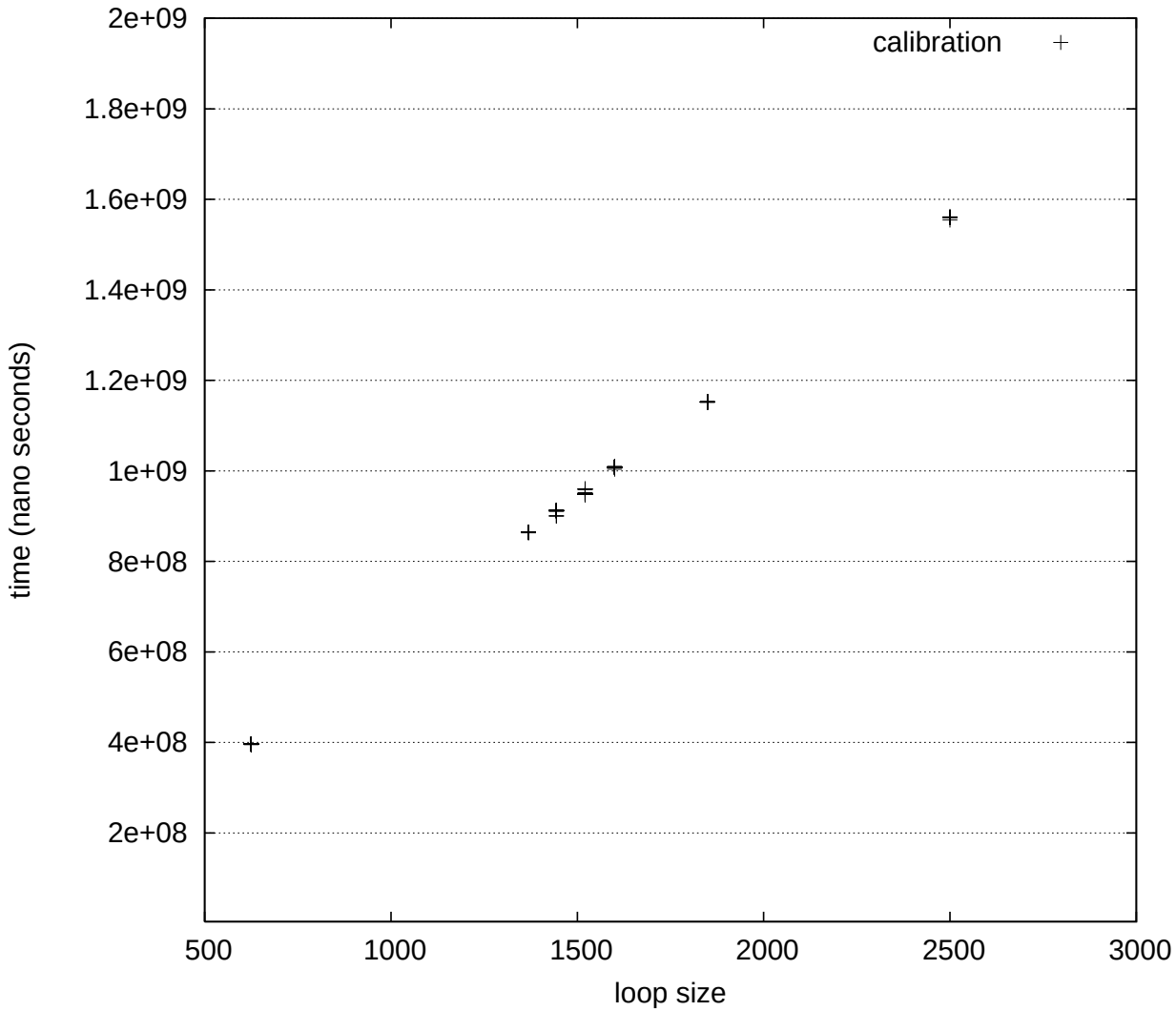
Passer à l'échelle: Moyenne



Écart type


std dev
based on nano second data
loop size
2e+07
1.8e+07
1.6e+07
1.4e+07
1.2e+07
1e+07
8e+06
6e+06
4e+06
2e+06
0
5000
10000
15000
20000
25000
30000

Recherche dichotomique



Isoler les mesures qui nous interressent

Pile	`.stack 2;` `.locals 0;`	`.stack 2;` `.locals 0;`	Pile

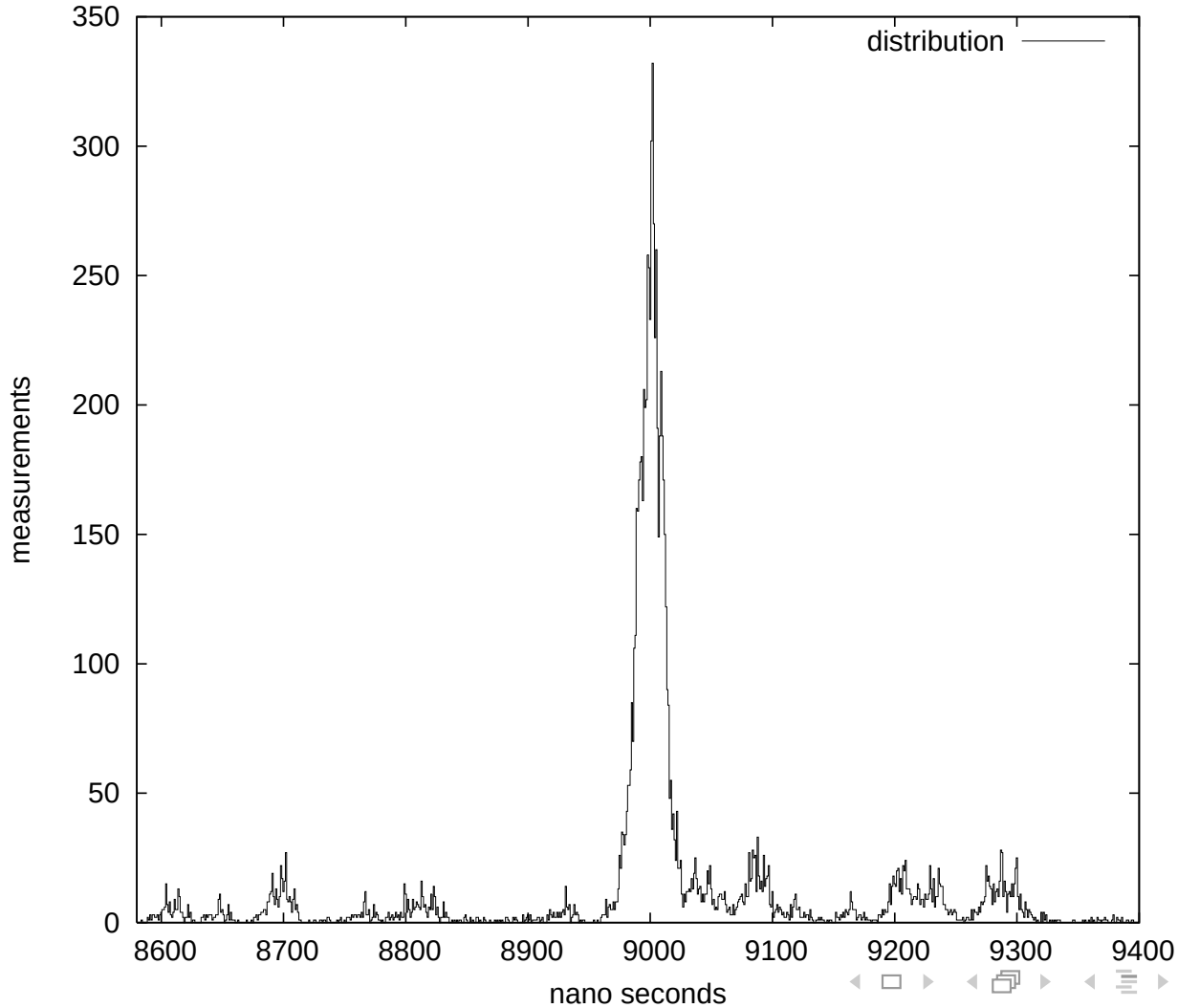
3
3;3
	`sconst_3;`
	`sconst_3;`
	`return;`
		`sconst_3;`
		`sconst_3;`
		`sadd;`
		`return;`
			3
			3;3
			6

Isoler les mesures qui nous interressent

Pile	`.stack 2;` `.locals 0;`	`.stack 2;` `.locals 0;`	Pile
3	`sconst_3;`	`sconst_3;`	3
3;3	`sconst_3;`	`sconst_3;`	3;3
6	`sadd;`	`sadd;`	6
	`return;`	`sconst_3;`	3;6
		`sadd;`	9
		`return;`	

Courbe de distribution - SADD, 10 000 valeurs

$\mu = 9013$ $\sigma = 131$



Les résultats à ce point

Nom	Unité mesurée	Temps moyen (ns)	Écart type	Taille de boucle
`ARITH16_REF`	`ARITH16_REF`	1.56E9	339033	2500
`SADD`	`ARITH16_REF` + 16×`SADD`	1.93E9	381054	2500

Résoudre quelques équations

Système de connaissance

$$loopsize \times (n_0 \times m_0 + n_1 \times m_1 + ... + n_p \times m_p) = t_0$$

$$loopsize \times m_i = t_i$$

Résoudre quelques équations

Système de connaissance

$$loopsize \times (n_0 \times m_0 + n_1 \times m_1 + ... + n_p \times m_p) = t_0$$

$$loopsize \times m_i = t_i$$

$$m_0 = \frac{\frac{t_0}{loopsize} - (n_1 \times m_1 + ... + n_p \times m_p)}{n_0}$$

Résoudre quelques équations

Système de connaissance

$$loopsize \times (n_0 \times m_0 + n_1 \times m_1 + ... + n_p \times m_p) = t_0$$

$$loopsize \times m_i = t_i$$

$$m_0 = \frac{\frac{t_0}{loopsize} - (n_1 \times m_1 + ... + n_p \times m_p)}{n_0}$$

$$2500 \times \texttt{ARITH16\_REF} = 1.56E9$$
$$2500 \times (16 \times \texttt{SADD} + \texttt{ARITH16\_REF}) = 1.93E9$$

Résoudre quelques équations

Système de connaissance

$$loopsize \times (n_0 \times m_0 + n_1 \times m_1 + ... + n_p \times m_p) = t_0$$

$$loopsize \times m_i = t_i$$

$$m_0 = \frac{\frac{t_0}{loopsize} - (n_1 \times m_1 + ... + n_p \times m_p)}{n_0}$$

$$\texttt{ARITH16\_REF} = 624000$$
$$\texttt{SADD} = 9250\text{ns}$$

Construire le système de coefficients

API - Paiement

API method	Nb	Tx
`getShort`	86	5
`arrayCopyNonAtomic`	72	0
`getBuffer`	60	0
`setShort`	40	2
`makeShort`	38	0
...		

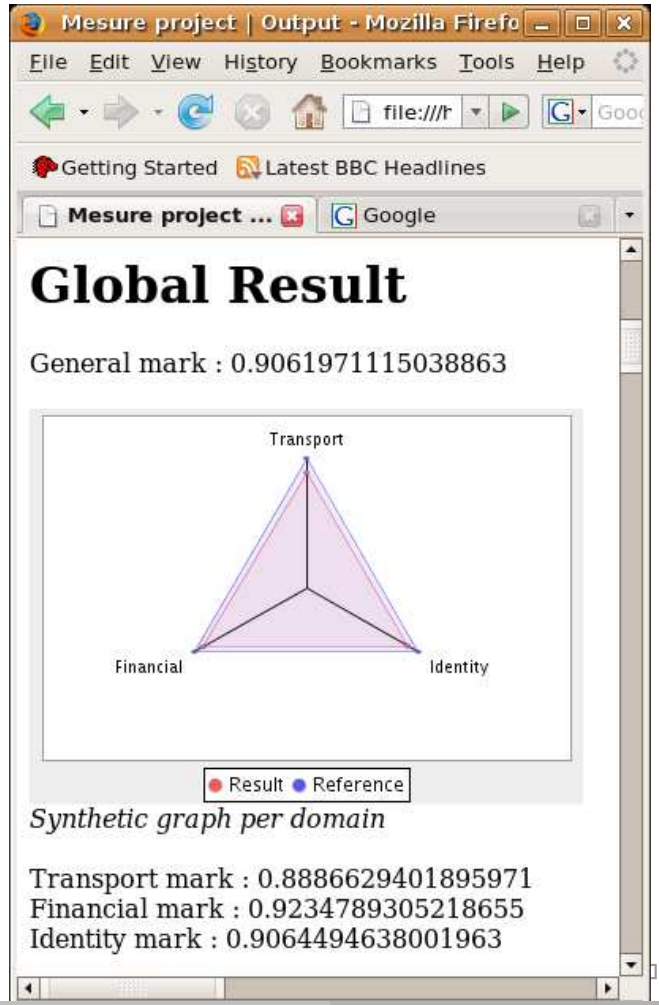
Construire le système de coefficients

Bytecodes - Transport

Bytecode	Nb	Tx
SLOAD	6106	0
SADD	3906	3
SSTORE	2738	0
SCONST_0	2562	2
GETFIELD_A_THIS	2514	0
...		

Note globale

On utilise les occurrences des bytecodes/entrées d'API pour établir des coefficients pour chaque domaine d'application.



Conclusion

- Java Card c'est historiquement l'essor des cartes à applications non dédiées
- Java Card 3.0 (specs en Mars 2008) : introduction du multithreading
- Un outil inhabituel pour les cartes : **open source**
- `http://mesure.gforge.inria.fr`
- Questions ?