The Structure and Performance of Interpreters

Jean-Loup Baer, Brian N. Bershad, Henry M. Levy,

Theodore H. Romer, Dennis Lee, Geoff Voelker, Alec Wolman, Wayne A. Wong

ASPLOS 1996 (p 150–159)

Architecture et performance des interpréteurs

Avantages des interpréteurs :

- vitesse de développement
- facilité d'utilisation
- portabilité
- sécurité

Mais :

- lenteur

Etude du comportement des interpréteurs et des possibilités d'améliorer leur performance

Description des interpréteurs

MIPSI

– émulateur binaire pour MIPS R3000
– outil pédagogique et de recherche
– le pipeline matériel est émulé en logiciel (étapes prefetch, decode, execute)

Java

– langage orienté objet
– utilisation d'un garbage collector
– librairies natives extensibles (ex : librairies graphiques)
– source compilé statiquement en bytecode
– bytecode interprété dynamiquement par la Java Virtual Machine (JVM)

Description des interpréteurs (suite)

Perl

- langage de script
- manipulation de texte, fichiers, processus
- source précompilé dynamiquement en une représentation interne
- représentation interne interprétée dynamiquement

Tcl

- langage de commande permettant l'interfaçage avec d'autre langages
- l'extension Tk permet le développement rapide d'interfaces graphiques
- source directement interprété

Architecture logicielle et performance des interpréteurs

– Vitesse des opérations simples
– Instructions virtuelles et instructions machine
– Modèle de gestion mémoire
– Performance d'applications réelles

Vitesse des opérations simples

Benchmark	Vitesse par rapport au C			
	MIPSI	Java	Perl	Tcl
a=b+c	260	100	770	6500
if then	80	20	190	1500
null procedure call	80	80	670	580
string-concat	190	500	20	80
string-split	65	160	13	30
file read	3	5	1	15

$\Longrightarrow$ importance des librairies natives

Instructions virtuelles et instructions machine

Machine virtuelle (MV)

– reconnaissance de l'instruction virtuelle,

– puis exécution des instructions machine correspondantes

Soit $ratio = \frac{nb_{instructions\ machine}}{nb_{instructions\ virtuelles}}$

Complexité de la MV	Décodage	$ratio$
simple (MIPSI, Java)	court	moyen
complexe (Perl, Tcl)	long	élevé

$\Longrightarrow$ le $ratio$ élevé compense le temps de décodage long

Temps de décodage très important pour Tcl

Modèle de gestion mémoire

	Nombre d'instructions machine par accès mémoire
MIPSI	60 - 62
Java	2 - 11
Perl	1 - 210
Tcl	200 - 500

Performance d'applications réelles

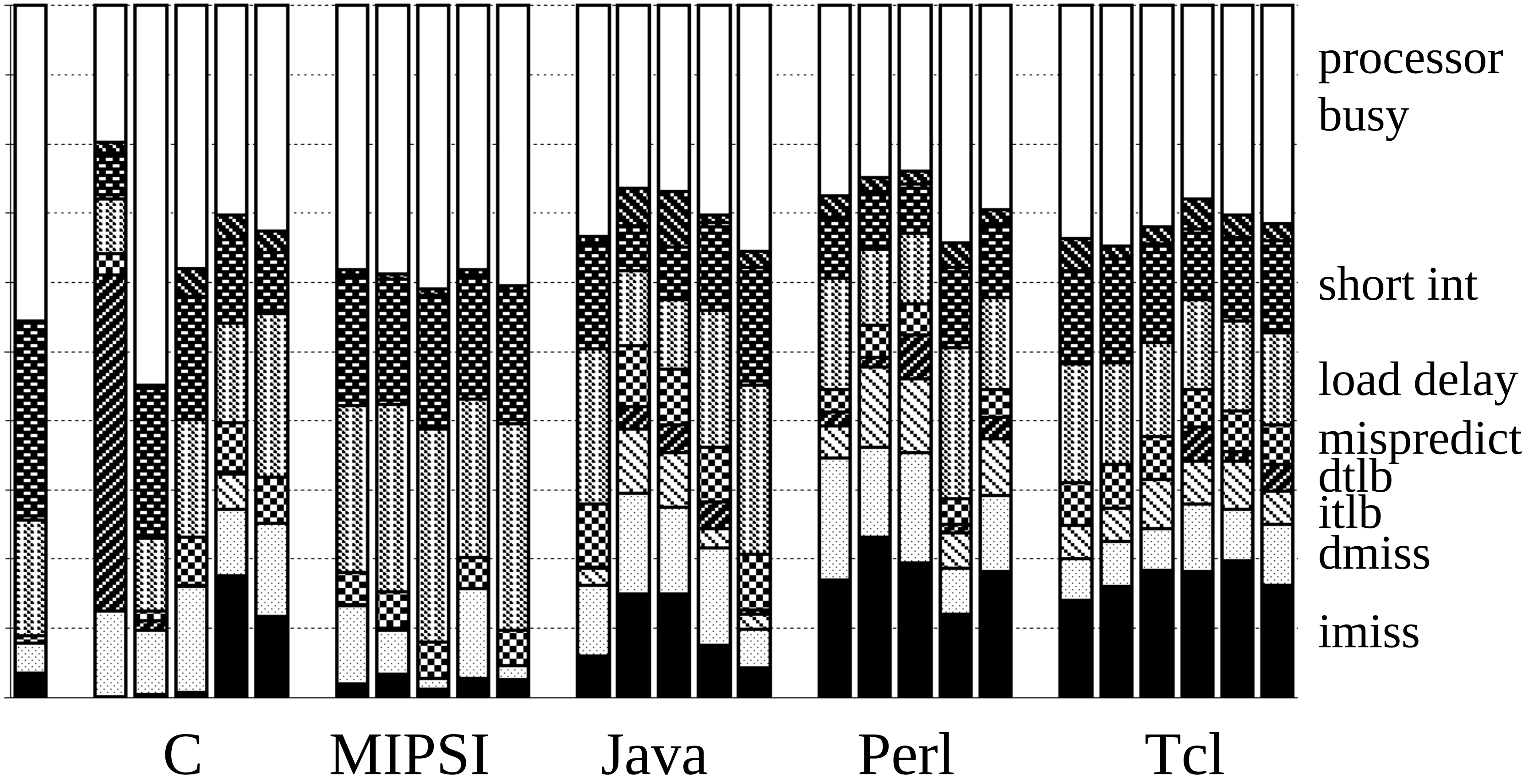
MIPSI : coût du décodage important

Java : importance des librairies natives

Perl & Tcl : la performance des applications dépend d'instructions virtuelles différentes

$\Longrightarrow$ pas d'instructions virtuelles particulières à optimiser

Comportement des interpréteurs vis-à-vis de l'architecture



Comportement global

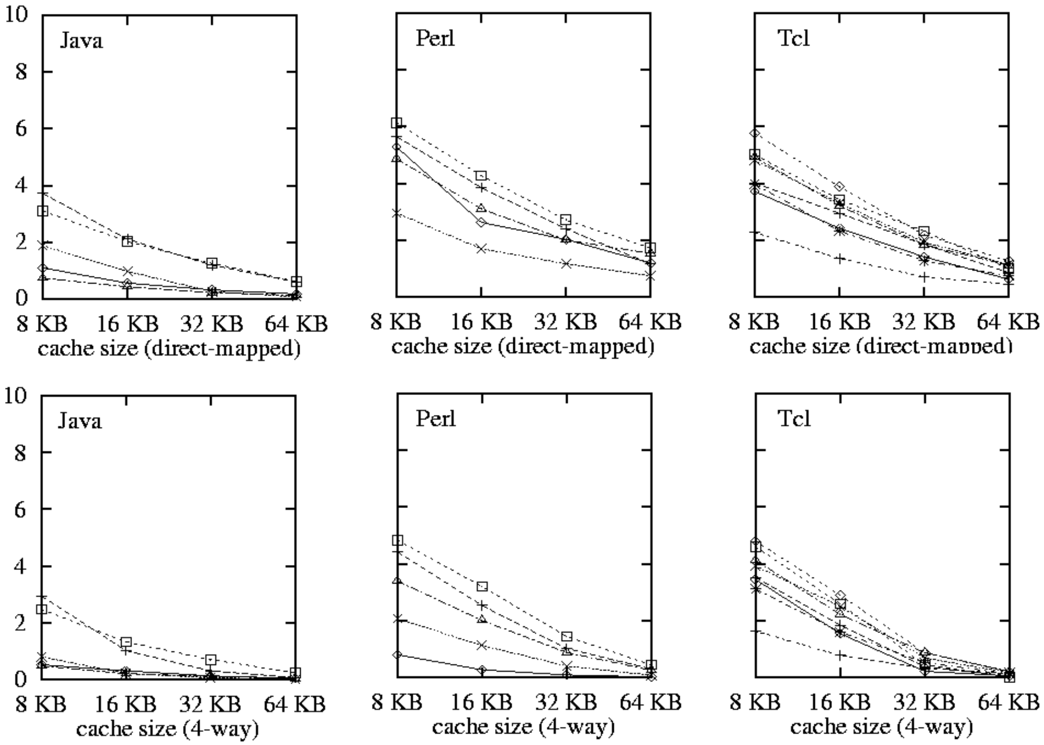
Le comportement de l'interpréteur l'emporte sur les spécificités des applications.

La performance ne dépend plus des applications.

– MV simple : bonne localité de données
– MV complexe : peu de localité sur les instructions et les données

Java = MV simple, mais librairies natives avec peu de localité

Quel cache d'instruction utiliser ?



– MV simple : 8 Ko direct-mapped
– MV complexe : 32 Ko associatif (éviter les conflict miss)

Conclusion

– importance des librairies natives
– complexité de la machine virtuelle
– comportement indépendant du programme