Un séjour chez Mozilla

Maxime Chevalier-Boisvert

En Bref

- Présentation de Higgs et Photon et rencontre de groupe du DLT
- Plusieurs rencontres individuelles avec Luke Wagner, Michael Bebenita, Shu-yu Guo, Jan de Mooij, Kannan Vijayan
- Invitations de Ariya Hidayat et Alex Gaynor
- Invitation à DConf 2013
- Visite du Computer History Museum avec Pierre Phaneuf
- Visite chez Apple, présentation de Higgs

Équipe JS de Mozilla

- Mon bureau était dans le coin de l'équipe JS
- ~10 personnes
  - ~6 au Mozilla HQ
  - +2 au Canada et en Europe
  - +Brian Hackett
- Ne travaillent pas tous sur le moteur JS en tout temps
- e.g.: Luke Wagner travaille principalement sur le support (et la spécification) asm.js

Moteur JS de Firefox

- Trois étages:
  - Interprète bytecode
  - Baseline JIT (JaegerMonkey)
  - Optimizing JIT (IonMonkey)
- La grande majorité du code n'est jamais JITté
  - 76% du code est mort
  - Grand pourcentage ne roule qu'un court temps
  - IonMonkey presque jamais utilisé sur le web
  - IonMonkey utilisé dans SunSpider et V8bench

Moteur JS: détails techniques

- IonMonkey utilise le stack C (sp)
- Aucun self-hosting, semble-t-il
  - map, forEach sont des fonctions C++
  - Routines assembleur par endroits (e.g.: toUint32)
- Heuristiques d'inlining complexes, “hand-tuned”
- Type specialization partiellement backporté dans JaegerMonkey
- Leur assembleur ne fait pas de sélection d'instruction ou d'autres optimisations

Les optimisations d'IonMonkey

- Type specialization
- Inlining
- Loop-invariant code motion
- Allocateur de registre linear scan
- Trucs spécializés:
  - Eval string cache
  - Memoization de fonctions trig. (SunSpider)
  - Backprop d'overflow entier (SunSpider)

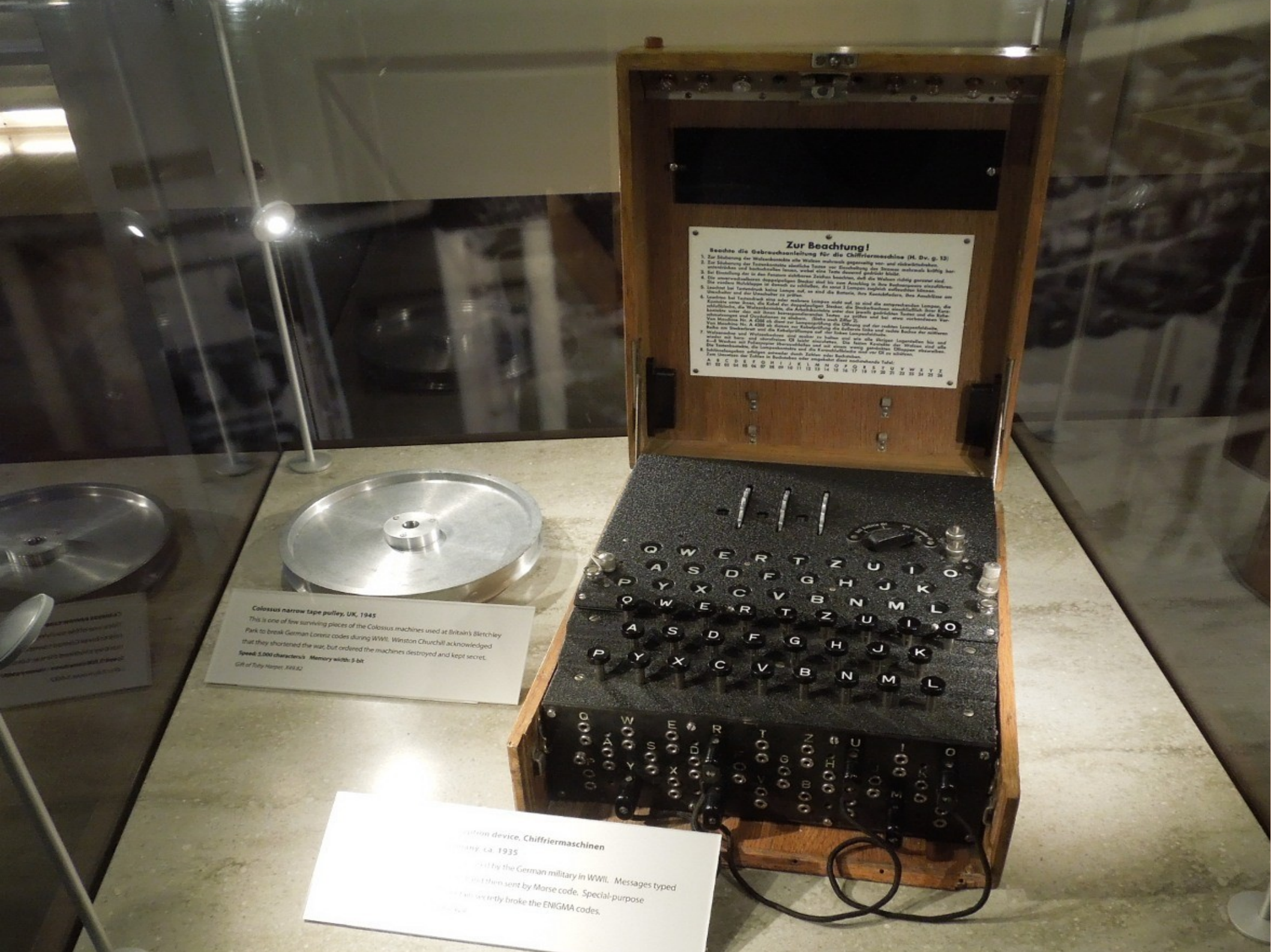
JS: Plans futurs de Mozilla

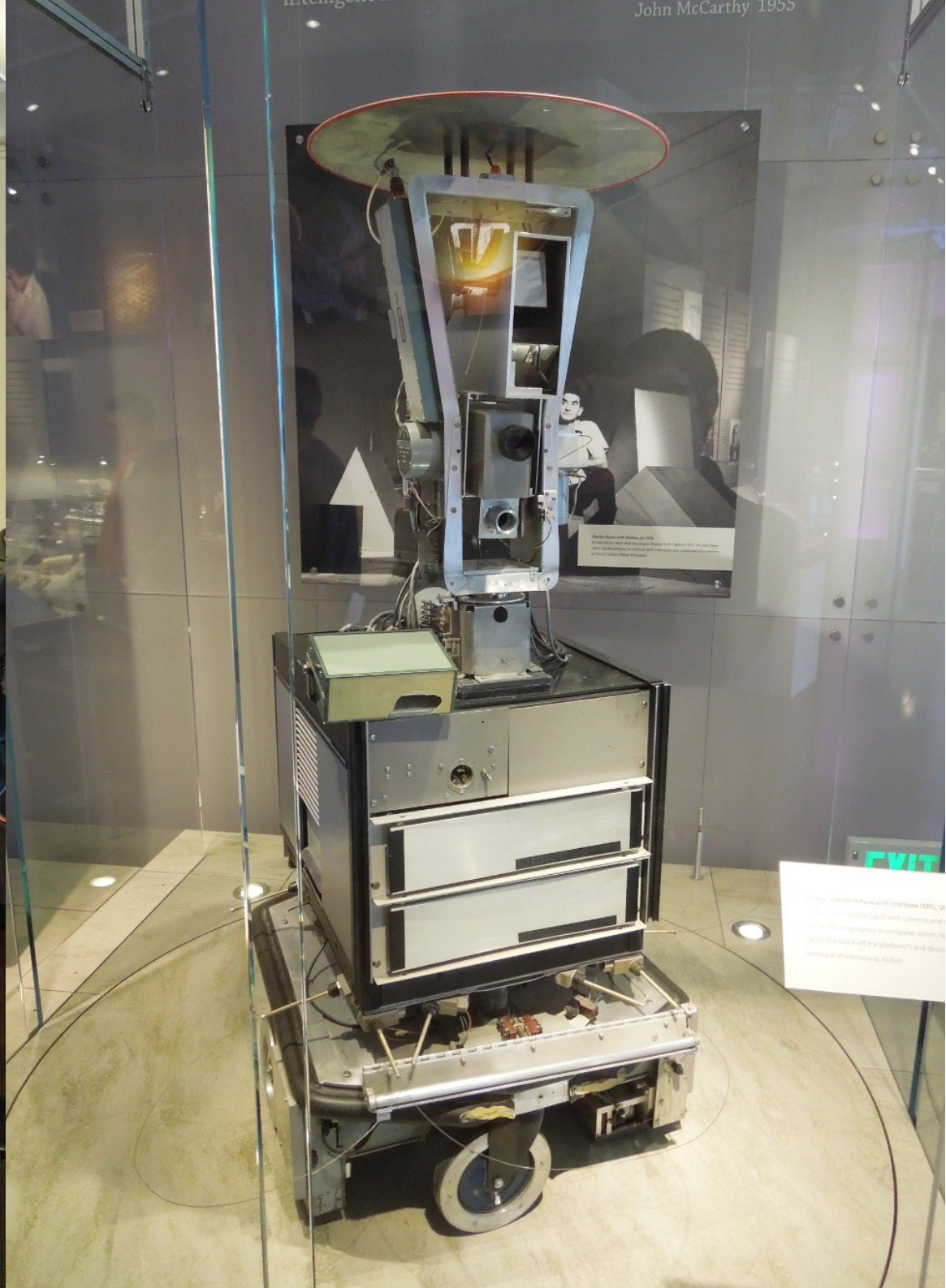
- Implantation d'un nouveau baseline JIT qui fait du monitoring de types (en cours)
  - Basé sur le codebase d'IonMonkey
- Remplacer l'interprète bytecode par un “AST walker”
  - “Interpreter speed doesn't matter at all”
- Éliminer le bytecode complètement
- Changer le format de boxing (inefficace pour pointeurs)
- Améliorations au type inference (encore indéfini)
  - Prise en compte du contexte

Type Inference

- Monitoring de types par les inline caches
  - getProp, arguments, valeur de retour
- Monitoring des overflows
- Chaîne de contraintes entre les instructions
  - Représentent la propagation des types
- Point-fixe interprocédural
- Maximum d'itération sur le point fixe
  - Si maximum atteint, guardes dynamiques insérés
- Pas de monitoring en putProp, getProp seulement
- Analyse de “definite properties” par pattern-matching sur les constructeurs


Computer History Museum


Zur Beachtung!
Colossus narrow tape pulley, UK, 1945
This is one of few surviving pieces of the Colossus machines used at Britain's Bletchley Park to break German Lorenz codes during WWII. Winston Churchill acknowledged that they shortened the war, but ordered the machines destroyed and kept secret.
Speed: 5,000 characters/s Memory width: 5-bit
Gift of Toby Harper. X4182
Chiffriermaschinen
by the German military in WWII. Messages typed
sent by Morse code. Special-purpose
secretly broke the ENIGMA codes.


John McCarthy 1955


BYTES FOR BITES:
THE KITCHEN COMPUTER
Why would anyone want a computer at home? Before the personal computer era and its avalanche of possible uses, the perennial answer was: "to store recipes."
Neiman-Marcus took that literally. The 1969 Christmas catalog featured the Kitchen Computer. For $10,600 you got the computer, a cookbook, an apron, and a two-week programming course.
Inside the futuristic packaging with a built-in cutting board was a standard Honeywell 316 minicomputer. But the console interface featured binary switches and lights. (Does 0011101000111001 mean broccoli? Or carrots?)
Not surprisingly, none were sold.
If she can only cook as well as Honeywell can compute.
Kitchen Computer, Neiman-Marcus/Honeywell, US, 1969


LISP Machine Inc.
LAMBDA
LMI

Chez Apple

- Donné talk similaire au talk chez Mozilla
- Réaction mixte, débat sur le système de boxing
- Brève discussion avec équipe JSC pendant lunch
- Accès interdit à leurs bureaux
- Culture beaucoup plus fermée que Mozilla

Double-word tagging

- Selon les gens de Mozilla, le JIT d'Opera (pas open source) utilise un stack de types séparés
- La performance de ce JIT était encore très compétitive avant l'arrivée de TI chez Mozilla

TraceMonkey

- Tracing JIT de mozilla, maintenant abandonné
- Problèmes techniques majeur
  - Incapable de ne pas spécialiser
  - Explosion des traces (chaque branche crée une nouvelle trace)
  - Énormes “kludges” pour gérer la récursion
  - Retombe à l'interprète très souvent
- Code “impossible à maintenir”
- Remplacé par JaegerMonkey (method JIT)

Shu et Luke sur le tracing

- Traces allow you to eliminate guards that were implied by previous checks
- “Tracing is all about tail duplication”
- "If you had two versions of a loop, a tracer could split them into two loops, but a method JIT couldn't"
- On TraceMonkey: “When it generates good code, it generated great code”
- “Tracing is too specialized"
- Luke: getProp with many possible classes, do you want to fork off many traces? Problematic with TraceMonkey
- When do you avoid specializing?

Alex Gaynor de PyPy

- Intéressé par l'aspect TI de Higgs
  - PyPy n'a rien de ce genre
- Sur l'explosion des traces:
  - PyPy a une cache de traces à taille limitée, jette du code s'il y en a trop
  - L'explosion des traces n'est pas un problème selon lui
- Pas de JS sur PyPy mature à sa conaissance
- (Opinion): on pourrait facilement leur botter le JIT
  - Pas de compétition sérieuse dans leur espace

Invitation à DConf 2013

- 1-3 Mai 2013, Menlo Park (bay area)
- Invited talk
- Slot de 60 minutes
- Utilisation du langage D dans l'implantation de Higgs, experience report
- Amplement de temps pour parler de Higgs lui-même
- Marc propose de visiter Mozilla en même temps

Sur GitHub

- 192 commits
- 35 personnes suivent maintenant Higgs
- Reçu quelques pull requests de John Colvin
  - Fan du langage D

Mon sujet de thèse

- Semble difficile de compétitionner avec Mozilla dans l'arène TI
- Mozilla semble avoir complètement abandonné le tracing pour le moment
- Problèmes non-résolus
- Idée: design d'un compilateur “par segment” qui essaie d'aller chercher les avantages d'un tracing JIT, sans les inconvénients
- Nécessite de faire du path-profiling

Recherche littéraire

- Dans le monde dynamique, rien d'autre que les traces de boucles ou les fonctions en terme d'unité de compilation
- “Region-based compilation”, venant des mondes VLIW, HPC
  - Création (par analyse statique ou profilage et heuristiques) d'unités de compilation qui croisent les frontières des fonctions
- Mots clés intéressants: superblocs, “straightening” de code
- Avantage dynamique: accumulation de guardes

Progrès sur Higgs

- Une énorme quantité de bug fixes
- Rajouté support pour *load*, JSON
  - Bruno: ~2 heures de travail
- Supporte maintenant quasiment tout SunSpider, la majorité de v8bench
- Commencé refactorings nécessaires pour JIT
- Commencé port de l'assembleur de Tachyon
- Plan actuel: implémenter un “tracelet JIT” pour pouvoir expérimenter

San Francisco

- Ville très diverse et colorée
- Rappelle un peu Montréal
  - Fait amusant: code régional 415
- Tout le monde est en technologie
- Beaux parcs (voir le Golden Gate park)
- Bon transports en commun
- Loyers très élevés (2 chambres: 2500-3500)
- Très hippie, ville d'origine des hipsters
- Les bars ferment avant 2AM :(
- Une expérience à vivre

Luke Wagner dit qu'on peut demander ce qu'on veut du Mozilla Gear Store, faire passer ça par Dave Herman, qui devrais nous l'envoyer gratuitement

https://wiki.mozilla.org/GearStore