Appendix B

Prediction Algorithm

This chapter describes an algorithm for predicting performance based on the families of graphs reported by the self-scaling benchmark.

This algorithm takes as input two items (all variables are in **bold**).

- The graphs from the self-scaling benchmark, that is, measured workloads for a number of focal points { focal point #0 to focal point #(**focalNum**-1) } each with a different focal point for uniqueBytes **focalPoint.uniqueBytes[i]**, but otherwise the same focal point **focal.sizeMean**, **focal.readFrac**, **focal.seqFrac**, **focal.processNum**.
- An I/O workload {**target.uniqueBytes**, **target.sizeMean**, **target.readFrac**, **target.seqFrac**, **target.processNum**} for which to predict performance.

This algorithm returns an estimate of the performance for the input workload.

(1) Choose the focal point upon which to predict performance for the target workload. Pick the focal point whose value of uniqueBytes has performance closest to the performance of **target.uniqueBytes**.

```
/* find the two focal points whose values of uniqueBytes
   form the tightest range which includes target.uniqueBytes */

focalLess = focal point with largest value of uniqueBytes
            less than target.uniqueBytes

focalMore = focal point with smallest value of uniqueBytes
            greater than target.uniqueBytes

/* pick the focal point whose value of uniqueBytes has
   performance closest to the performance of target.uniqueBytes
   */

/* performanceLess is the performance at focal point focalLess */
performanceLess = performance of workload
            (focal.uniqueBytes[focalLess], focal.sizeMean,
            focal.readFrac, focal.seqFrac, focal.processNum)

/* performanceMore is the performance at focal point focalMore
   */

performanceMore = performance of workload
            (focal.uniqueBytes[focalMore], focal.sizeMean,
            focal.readFrac, focal.seqFrac, focal.processNum)

/* performance is the performance of the workload which is
   the same as the focal point except for uniqueBytes,
   which is equal to target.uniqueBytes. */
performance = performance of workload (target.uniqueBytes,
     focal.sizeMean, focal.readFrac, focal.seqFrac, focal.processNum)

if (performance is closer to performanceLess) {
     focalUse = focalLess
} else {
     focalUse = focalMore
}

performance = performance at focal point focalUse
```

(2) Use focal point **focalUse** to adjust **performance** for differences between the focal point and the target workload, one parameter at a time.

```
for each parameter (uniqueBytes, sizeMean, readFrac, seqFrac,
          processNum)
      /* adjust for parameter */

      /* performance1 is the performance at focal point focalUse
         */

      performance1 = performance of workload
            (focal.uniqueBytes[focalUse], focal.sizeMean, focal.readFrac,
            focal.seqFrac, focal.processNum)

      /* performance2 is the performance of the workload which
         is the same as the focal point except for parameter,
         which has the value of the workload whose perfor-
         mance is being predicted.  Here I've shown the
         workload assuming parameter is sizeMean */

      performance2 = performance of workload
            (focal.uniqueBytes[focalUse], target.sizeMean, focal.readFrac,
            focal.seqFrac, focal.processNum)

          performance *= performance2 / performance1
end for

return(performance)
```